A GENERATIVE THEORY OF RELEVANCE

A Dissertation Presented

by

VICTOR LAVRENKO

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

September 2004

Computer Science

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
SEP 2004		**-**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
A Generative Theory of Relevance	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Space and Naval Warfare Systems Center,53560 Hull Street,San Diego,CA,92152-5001**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
The original document contains color images.

14. ABSTRACT
see report

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**		**130**	

To Daria

ACKNOWLEDGMENTS

This thesis would not have been possible without the help of my relatives, mentors and colleagues. First and foremost, I am thankful to my parents for igniting in me the passion for scientific discovery, for teaching me how to think and how to wonder. I was lucky to have a father who could explain trigonometry to a ten year old, and a mother who taught me to strive for perfection. I was also fortunate to have grandparents who served as role models of determination and hard work. I am truly indebted to John Stuber; without his kindness my life would have taken a very different path. I want to give special thanks to my wife for her understanding, patience and unending support during these difficult years.

Having two academic advisors is a challenging but fascinating experience. I was fortunate to work closely with Bruce Croft, whose knowledge of the field is truly astounding, and who showed me the importance of seeing the wood behind the trees. I am also tremendously thankful to James Allan, who guided me since the early days of my undergraduate studies. I am particularly grateful to James for his willingness to give unbiased advice, even when it may have run contrary to his interests.

I would like to thank Louise Knouse for turning the dry art of abstract mathematics into an accessible and fun discipline. I am also grateful to Donald Geman and Alexey Koloydenko for showing me the power and the awe-inspiring beauty of probability theory. I am thankful to Sherre Myers for proofreading this work.

Finally, I would like to thank all of my fellow students and colleagues at the Center for Intelligent Information Retrieval. Early discussions with Warren Greiff and Anton Leouski had a strong impact on my way of thinking about the field of information retrieval. I greatly enjoyed working with Jeremy Pickens – it was a rare case of complementary skills leading to a very fruitful collaboration. I fondly remember the many absorbing discussions I shared with Dawn Lawrie, Nicola Stokes and Ramesh Nallapati. I would like to give special thanks to Andres Corrada-Emmanuel; his unorthodox point of view and willingness to challenge assumptions played an invaluable part in helping me sort out and solidify many of the ideas presented in this thesis.

This material is based on work supported in part by the National Science Foundation, Library of Congress and Department of Commerce under cooperative agreement number EEC-9209623, in part by Dragon Systems, Inc. and Space and Naval Warfare Systems Center, San Diego under cooperative agreement #N66001-00-2-8929 and in part by SPAWARSYSCEN-SD grant number N66001-02-1-8903. Any opinions, findings and conclusions or recommendations expressed in this material are the author(s) and do not necessarily reflect those of the sponsor.

ABSTRACT

A GENERATIVE THEORY OF RELEVANCE

SEPTEMBER 2004

VICTOR LAVRENKO

B.Sc., UNIVERSITY OF MASSACHUSETTS AMHERST

M.Sc., UNIVERSITY OF MASSACHUSETTS AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor W. Bruce Croft and Professor James Allan

We present a new theory of relevance for the field of Information Retrieval. Relevance is viewed as a generative process, and we hypothesize that both user queries and relevant documents represent random observations from that process. Based on this view, we develop a formal retrieval model that has direct applications to a wide range of search scenarios. The new model substantially outperforms strong baselines on the tasks of ad-hoc retrieval, cross-language retrieval, handwriting retrieval, automatic image annotation, video retrieval, and topic detection and tracking. Empirical success of our approach is due to a new technique we propose for modeling exchangeable sequences of discrete random variables. The new technique represents an attractive counterpart to existing formulations, such as multinomial mixtures, pLSI and LDA: it is effective, easy to train, and makes no assumptions about the geometric structure of the data.

TABLE OF CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

Information Retrieval (IR) is a field of science concerned with searching for useful information in large, loosely structured or unstructured collections. In the 1970s, when the field was in its infancy, the word *information* primarily meant *text*, specifically the kinds of text one might find in a library. Today information exists in many more forms, often quite different from books or scientific articles which were the focus of early retrieval systems. Web searching is perhaps the most famed application of information retrieval today, and on the web we may find relevant information in the form of text, but also as images, as audio files, as video segments, even as hyper-links between various pages. What brings together these very different manifestations of information is their *relevance* to what the user is searching for.

The notion of relevance plays an extremely important part in Information Retrieval, some went so far as to say that it defines the entire field and serves as a distinguishing feature from documentation or library science [116]. The concept of relevance, while seemingly intuitive, is nevertheless quite hard to define, and even harder to model in a formal fashion, as evidenced by over 160 publications attempting to deal with the issue. Relevance is also the subject of this thesis. We will not attempt to bring forth a new definition of relevance, nor will we provide arguments as to why a new definition might be theoretically superior or more complete than what has been proposed previously. Instead, we will take a widely accepted, albeit somewhat conservative definition, make several assumptions, and from them develop a new probabilistic model that explicitly captures the notion of relevance without the need for heuristics.

1.1 Contributions

There are two major contributions in this thesis. The first contribution is a new way to look at topical relevance within the field of Information Retrieval. The second contribution is a new generative model for collections of discrete data.

1.1.1 A new model of relevance

Our first contribution will be of primary interest to a researcher in Information Retrieval, or to anyone concerned with modeling topical content. This dissertation will propose a new way to look at topical relevance. The new approach will be summarized as a hypothesis (defined in section 3.1), and will lead to a new formal model of Information Retrieval. Some of the main advantages of the new model are:

- It represents a natural complement of the two dominant models of retrieval: the classical probabilistic model and the language modeling approach.
- The model *explicitly* combines documents, queries and relevance in a single formalism. This has been somewhat of a challenge in previous formulations.
- It requires no heuristics to compute the probabilities associated with the relevant class.
- The retrieval model has relevance feedback and query expansion as integral parts.
- It captures a wide array of IR tasks. As far as we know, this is the first model capable of performing ad-hoc search, image and video annotation, handwriting retrieval and Topic Detection all within the same formalism.

- It outperforms state-of-the-art benchmarks. A few examples: 10-25% improvement in ad-hoc search, 5-30% in topic detection, 15-60% in image annotation and retrieval.
- It allows queries to be issued in any form: keywords, passages, whole documents, sketches, portions of images, video frames – as long as we have the data to train it.
- It can recover missing portions of incomplete or degraded information representations.

1.1.2 A new generative model

The second contribution of our thesis may be useful to a language-modeling practitioner, or to a reader with a more general interest in statistical modeling (not necessarily of language). In chapter 4 we will propose a new formalism for modeling exchangeable sequences of discrete random variables. The main distinguishing qualities of our formalism are:

- It can successfully handle rare events and does not ignore outliers in the training data.
- It does not make any structural assumptions, allowing the data to speak for itself.
- The model is easy to train, either generatively, or with a discriminative objective.
- The model is effective: 15-20% reduction in predictive perplexity over the baselines, plus successful applications to retrieval tasks mentioned in the previous section.

1.1.3 Minor contributions

In addition to the two major contributions, this thesis contains a number of smaller results. These results mostly represent new insights into existing models, or serve as explanations for curious experimental observations. Some of these minor contributions are:

- We will show that, contrary to popular belief, the classical probabilistic model is not based on the assumption of word independence. Instead, it requires a much more plausible assumption of *proportional interdependence*.
- We will try to explain why explicit models of word dependence have never resulted in consistent improvements in retrieval performance, either in the classical or the language-modeling framework.
- We will assert that documents and queries can be viewed as samples from the same underlying distribution, even though they may look very different.
- We will argue that document likelihood, or probability ratio, is not the best criterion for ranking documents. We will contend that query likelihood is a better alternative.
- We will conclude that a Probabilistic LSI model with k aspects is equivalent to a regular, document-level mixture model with $N >> k$ components.
- We will argue that Latent Dirichlet Allocation is not a word-level mixture, it is a regular document-level mixture restricted to the k-topic sub-simplex.

1.2 Overview of the Thesis

The remainder of this thesis will be structured in the following manner. **Chapter 2** will be devoted to the notion of relevance and to previously proposed models of relevance. We will start with a simple definition and then briefly outline several arguments challenging that definition. Sections 2.1.2 through 2.1.5 will contain several alternative views of relevance. Conceptual discussions will culminate in section 2.2, where we will describe Mizarro's attempt [89] to construct a unified definition of relevance. In section 2.3 we will discuss two main approaches for dealing with relevance: the classical probabilistic framework (section 2.3.2) and the

language modeling framework (section 2.3.3). Our discussion will focus heavily on the probabilistic representations adopted by the two models, and on the modeling assumptions made in each case. Sections 2.3.2.5 and 2.3.3.4 contain novel arguments suggesting that modeling word dependence is a futile endeavor, at least if one is concerned with topical relevance. We conclude chapter 2 by contrasting the two frameworks and pointing out that both are not completely adequate: the classical framework because of its reliance on heuristics, and the language modeling framework because of the explicit absence of relevance in the model.

Chapter 3 contains the main theoretical ideas of the thesis. In this chapter we will introduce a new framework for viewing relevance and describe how this framework can be applied in a number of search scenarios. In section 3.1 we will provide an informal overview of our ideas. We will also define the core proposition of our work: the *generative relevance hypothesis*. Section 3.2 will contain a formal development of our framework. Section 3.2.1 will focus on representation of documents, queries and relevance, as well as on functional relations between these representations. Section 3.2.2 will discuss probability distributions associated with relevance, and how these distributions affect document and query observations. In section 3.2.3 we describe how our model can be used for ranking documents in response to a query. We will discuss three existing ranking criteria: document likelihood, query likelihood and model comparison. We will also discuss relations between these criteria in section 3.2.3.4 and provide an original argument for why document likelihood may lead to sub-optimal document ranking (section 3.2.3.5). Section 3.3 concludes the chapter and contrasts our model against both the classical probabilistic model and the language modeling framework.

Chapter 4 is dedicated to the statistical machinery underlying our model. The methods discussed in this chapter are very general and extend well beyond the scope of this thesis. We consider a deceivingly simple problem of modeling exchangeable sequences of discrete random variables, commonly known as *bags of words*. In section 4.2 we review five existing approaches to this problem, starting with a simple unigram model and progressing to advanced models like PLSI and Latent Dirichlet Allocation (LDA). In section 4.3 we demonstrate that all five models can be expressed in a simple common form that is a direct consequence of De-Finetti's representation theorem. We draw several unexpected conclusions, showing that pLSI is equivalent to a simple discrete mixture, and LDA is a restricted form of a simple continuous mixture. Section 4.4 will introduce a completely new family of generative models, predicated on kernel-based density allocation within De-Finetti's representation. We will discuss two types of kernels: Dirac delta functions and Dirichlet kernels. We will also give an extensive argument for why we believe the new family is superior to existing formulations.

Chapter 5 takes the general ideas from chapters 3 and 4 and shows how they can be turned into operational search systems. We discuss application of our framework to seven very different retrieval scenarios:

1. Ad-hoc retrieval: similar to simple web search, minus the hyperlink information.
2. Relevance feedback: use interaction with the user to improve searching accuracy.
3. Cross-language retrieval: use English queries to find documents in Chinese.
4. Handwriting retrieval: search through highly degraded historical manuscripts
5. Image annotation: automatically annotate images with related keywords
6. Video retrieval: search through a video collection using a text query
7. Topic Detection and Tracking: organize live news-feeds into event-based groups

In each case we will start by describing how the scenario fits into our model of relevance. We will then provide a detailed empirical evaluation, showing the effectiveness of our model against state-of-the-art baselines.

Chapter 6 will conclude the thesis. We will discuss the implications of our experiments and make suggestions for further development of our framework.

CHAPTER 2

RELEVANCE

2.1 The many faces of relevance.

The notion of relevance serves as the foundation for the field of Information Retrieval. After all, the purpose of retrieval systems is to retrieve relevant items in response to user requests. Naturally, most users (with a few exceptions) have a fairly good idea of what relevance is – *it is that thing they are looking for*. However, in order to build and test effective retrieval systems we must formalize the concept of relevance, and that turns out to be somewhat of a challenge.

2.1.1 A simple definition of relevance

One of the simplest (and also most widely used) definitions of relevance is that of a binary relation between a given information item (document D) and the user's request (query Q). We might assume that the document is represented by a set of key words, appropriately reflecting its contents. Similarly, the user's request Q is a set of key words that represent the user's interest. The simplest definition of relevance is that D is relevant to Q if there is a substantial semantic overlap between the representations of D and Q. Relevance of D does not depend on any factors other than representations of D and Q. Specifically, it does not depend on the user who issued the request, the task that prompted the request, or on user's preferences and prior knowledge. Similarly, in this simple definition relevance does not depend on any other documents D' in the collection, whether or not they have been examined by the user, or even judged relevant or non-relevant. It also does not depend on any other requests Q' to which D was previously judged relevant or non-relevant.

When relevance is defined as above, it is often called *system-oriented* or *algorithmic* relevance. The definition has been challenged and deemed inadequate on numerous occasions. In his comprehensive review of various formulations of relevance Mizarro [88] cites 160 papers attempting to define various aspects of relevance. In this section we will briefly highlight a few of the most popular arguments about the nature of relevance. A reader yearning for a more complete exposition of the subject is invited to examine the work of Saracevic [116], Robertson [109], Harter [53] and Mizarro [88].

2.1.2 User-oriented views of relevance

Discussions of the proper definition of relevance began in 1959 when Vickery[139, 140] argued for a distinction between "relevance to a subject" and "user relevance". The former refers to a degree of semantic correspondence between the user's request and an item returned by the system in response to that request. The latter is a reflection of how much the user likes an item, taking into account his task and previously seen items. The distinction between the two is best explained if we imagine that our collection contains two near duplicates of the same item. If one of them is semantically relevant to the request, so will be the other. However, the user may not be satisfied with the second item, since it is completely redundant.

A somewhat different dichotomy of relevance was mentioned by Maron and Kuhns, who incidentally were the first to treat relevance probabilistically. In [82] they consider the distinction between the user's request and the underlying *information need*. The request is a surface representation of information need, it is observable and readily available to the system and to other users. The information need itself is an abstract concept that only the user himself is aware of. A document that appears relevant to the request may be completely irrelevant to the underlying information need. This happens frequently because of ambiguity inherent in human communication; the problem is further aggravated by the users' tendency to keep requests short and devoid of little redundancies that would be so helpful to a retrieval engine.

Belkin et al. [9, 10] make a further abstraction of a user's *information need*. They introduce a concept of ASK (*Anomalous State of Knowledge*), signifying the fact that the user himself may not be fully aware of what he is searching for. The concept of information need is completely abstract, it is not observable; the user has only a perception of that need, and that perception can change during the course of a searching session. The same idea is raised in a number of other publications. For example, Ingwersen [59], coins the acronyms ISK and USK, referring respectively to *Incomplete* and *Uncertain* States of Knowledge.

Foskett [46, 47] and later Lancaster [71] make an interesting argument that the nature of relevance to a large degree depends on the person who is making a judgment. In their definition, the term *relevance* refers to a "public" or "social" notion, where the judgment is made by an external expert or collective, and not by the user who posed the request in the first place. For the case when the judgment is made by the user himself, they coin the term *pertinence*. In light of previous discussion by Maron and Kuhns [82], relevance is a relation between a document and the request, while pertinence is a relation between a document and the underlying information need.

In opposition to most of the distinctions drawn above, Fairthorne[43] makes a sobering argument for a strict definition of relevance, involving only the words contained in the document and the query. Otherwise, he claims, for any given document and any request we could hypothesize a situation where that document would in fact be relevant to the request.

2.1.3 Logical views of relevance

A number of authors attempted to define relevance formally, through logical constructs on the semantics of the documents and requests. One of the first formal attempts is due to Cooper [25], who defines relevance in terms of entailment (as in theorem-proving). Suppose q represents a logical proposition corresponding to a user's request. Let s be a proposition reflecting the meaning of some given sentence. Cooper says that s is relevant to q if s is a necessary assumption one needs to make in order to prove q. Formally, it means that s belongs to a minimal set S of propositions that entail q:

$$rel(s,q) \iff \exists S : s \in S, S \models q, S - s \not\models q$$

A document D is considered relevant to the request if it contains at least one sentence s that is relevant to q. Cooper's definition is quite attractive in that it allows relevance to be defined on a sub-document level. This makes it reasonable to allow that documents may discuss different topics and that a single document may be relevant to very different requests. The definition also allows relevance to be extended into *novel* or *marginal* relevance, which will be discussed later in this section.

There are two main criticisms of Cooper's definition of relevance. One is operational, and has to do with the inherent difficulty of transforming natural language into logical propositions. The other is conceptual – Cooper's relevance is a binary relation between a query (expressed in logical form) and a document. The user is not in the picture, and neither is there any way to factor in the task that prompted the search in the first place. Consequently, with Cooper's definition a document ought to be judged relevant by every user that happens to generate q as their request, regardless of the task they are faced with or their personal preferences. An attempt to improve Cooper's definition was made by Wilson [146], who introduced the idea of *situational relevance*. Wilson's concept of relevance includes the situation in which the search is performed, user's goals, as well as the information already known to the user prior to examining a given document.

Cooper's definition of relevance experienced a strong revival when it was re-formulated in 1986 by Van Rijsbergen [134]. Van Rijsbergen's work was followed by a large number of publications describing relevance in terms of various formal logics [94, 95, 96, 97, 84, 118, 31, 32]. One important characteristic of most, if not all of these re-formulations is that they try to replace strict logical entailment ($d \models q$) with some weaker, but more tractable form. For example Van Rijsbergen [134, 135, 137] replaces the strict entailment with *logical implication* (denoted $d \rightarrow q$). Bruza [19, 20, 18] introduces a concept of *plausible entailment* and argues that a document d should be considered relevant as long as the query is at least plausibly entailed by some part of d. Lalmas and Van Rijsbergen [68, 69, 70] use modal logic and situation theory, treating a document as a *situation* and declaring it relevant if a flow of information from a document may lead to another situation, which in turn could strictly entail the query.

2.1.4 The binary nature of relevance

In most definitions relevance takes a form of a binary relation between a document and a query – a document is either relevant or it is not. A number of authors attempted to introduce graded notions of relevance, either in terms of discrete categories [38, 39, 65], or in terms of points or confidence intervals on the real line [42, 60]. Some authors also prefer to re-cast graded relevance as *utility* [28], or as a decision-theoretic notion of *risk* [153]. While interesting in their own right, these definitions usually run into a number of practical difficulties. It turns out that graded relevance judgments are more costly to obtain, and the agreement between different judges is lower than for binary relevance. In addition, there is some evidence that human judges naturally prefer to use the end points of a given scale, hinting at the dichotomous nature of relevance [61]. Also, the methodology for evaluating retrieval effectiveness with graded relevance is not nearly as developed as it is for the case of binary relevance, where there is a small set of almost universally accepted performance measures.

2.1.5 Dependent and independent relevance

Most operational systems assume that relevance of a given document is independent of any other document already examined by the user, or of any other unexamined document in the collection. The assumption is motivated by a number of practical concerns. First, non-independent relevance judgments are considerably more expensive to obtain, since the judgment for a particular document will depend on the order in which all documents are presented to the judge. Second, retrieval algorithms themselves become computationally expensive when we have to search over the subsets of a large collection of documents. However, assuming independent relevance often takes us too far from the reality of a information seeking process, so a number of alternative definitions exist and will be outlined below.

2.1.5.1 Relevance and novelty

Redundancy of information is perhaps the most common reason for considering alternative, non-independent definitions of relevance. If a collection contains two near-duplicate documents the user is unlikely to be interested in reading both of them. They both may be topically relevant to the request, but once one of them is discovered, the second one may become entirely redundant, and irrelevant for information seeking purposes. To reflect the value of novelty, Carbonell and Goldstein [21] proposed the concept of *maximal marginal relevance* (MMR). Their idea was to provide a balance between the topical relevance of a document to the user's request, and redundancy of that document with respect to all documents already examined by the user. Allan and colleagues [6, 4] recognized the fact that novelty and redundancy must also be addressed on a sub-document level. For example, a document may be mostly redundant, but may contain a small amount of novel and very pertinent information. Something like this often happens in news reporting, where journalists tend to re-use a lot of previously reported information, interspersing it with important new developments. In Allan's formulation there are two separate definitions of relevance – topical relevance and novel relevance, and system performance is evaluated independently with respect to both of them. The importance of novelty in ranking has finally lead to the establishment of the *Novelty Track* within the Text Retrieval Conference (TREC) [49], which has attracted a large number of publications in the last two years.

2.1.5.2 Relevance of a set of documents

If the information need of the user is sufficiently complex, it may be possible that no individual document completely satisfies that need by itself. However, information pooled from several documents may be sufficient. In this case the assumption of independent relevance clearly does not hold and we have to conceptualize relevance of a *set of documents* to the information need. For a single document, we may define a notion of *conditional* relevance, where conditioning is with respect to other documents included in the retrieved set. One of the first formal models of conditional relevance is due to Goffman [48], who defines the relevant set as a *communication chain* – a closed sequence of documents where relevance of a given document is conditioned on the previous document in the sequence (and must exceed a certain threshold for the document to be included). Goffman's model was evaluated by Croft and Van Rijsbergen [36] with results suggesting that

Goffman's model, while considerably more expensive from a computational standpoint, was not superior to simpler forms of ranking.

A major step away from independent document relevance was taken by Van Rijsbergen when he defined the now famous *cluster hypothesis*[133]: "Documents that cluster together tend to be relevant to the same requests." The hypothesis was tested in a large number of studies, notably [62, 138, 141, 55, 79]. However, with few exceptions all these studies evaluate relevance at the level of individual documents in a cluster, rather than relevance of the cluster as a whole. Clustering was used primarily as a different way to organize retrieved documents.

2.1.5.3 Aspect relevance

In some cases, a complex information need can be broken down into smaller independent components. These components are often called *aspects*, and the goal of the retrieval system is to produce a set of documents that cover as many aspects of the overall need as possible. In this setting, it is common to define *aspect relevance* as topical relevance of a single document to a particular aspect of the overall need, and *aspect coverage* refers to the number of aspects for which relevant documents exist in the retrieved set. Aspect relevance and aspect coverage have been extensively studied in the Interactive Track of the TREC conference. One formal approach to modeling aspect relevance is described by Zhai [153].

2.2 Attempts to Construct a Unified Definition of Relevance

Faced with a growing number of definitions of relevance, several authors attempted to come up with a unified definition, which would classify and relate various notions of relevance. Some of the more prominent attempts were made by Saracevic [117], Mizarro [88, 89] and Ingwersen [30]. In this section we will briefly describe one particularly interesting proposal to formalize relevance by embedding it in a partially ordered four-dimensional space.

According to Mizarro [89], almost any reasonable definition of relevance can be represented as a vector consisting of four components: {Information,Request,Time,Components}. The four dimensions of Mizarro's space have the following interpretations:

1. **Information type.** The first dimension of relevance is the kind of information resource for which we are defining relevance. In Mizarro's definition, this dimension can take one of three values: *document*, *surrogate* and *information*. Here *document* refers to the physical item a user will receive as the result of searching – the full text of a document, or, in the case of multi-media retrieval, a complete image, a full audio or video, file. *Surrogate* refers to a condensed representation of an information item, such as a list of keywords, an abstract, a title, or a caption. *Information* refers to changes in the user's state of knowledge as a result of reading or otherwise consuming the contents of a document. Note that information is a rather abstract concept, it depends on user's state of knowledge, his attentiveness, his capacity to comprehend the contents of the document and an array of other factors.

2. **Request type.** The second dimension of relevance specifies a level at which we are dealing with the user's problem. Mizarro defines four possible levels: *RIN*, *PIN*, *request* and *query*. The first (*RIN*) stands for Real Information Need and defines the information that will truly help the user solve the problem that prompted him to carry out a search in the first place. Needless to say, the user may not even be fully aware of what constitutes his real information need, instead he *perceives* it, and forms a mental image of it. That image is called *PIN*, or Perceived Information Need. Once the user knows (or rather thinks that he knows) what he is searching for, he formulates a *request*. A request is a natural language specification of what the user wants to find, something that might be given to a knowledgeable librarian or an expert in the field. A request is a way of communicating the *PIN* to another human being. Finally, this request has to be turned into a *query*, which is something that can be recognized by a search engine, perhaps a list of key words, a boolean expression or an SQL query. A query is a way of communicating the request to a machine.

3. **Time.** The third dimension of relevance reflects the fact that searching is not a one-shot process. The user may not see any relevant items in the initial retrieved set, and this may prompt him to re-formulate

the *query*, perhaps changing the boolean structure, or adding some additional keywords. If the user does find relevant items, information in these items may prompt him to formulate different *requests*, or perhaps even force him to re-think what it is he wants to find, thus changing the perception of his information need (*PIN*). The real information need (*RIN*) stays constant and unchanging, since it is refers to what the user will ultimately be satisfied with. Mizarro endows the third dimension of relevance with a discrete progression of time points: $\{i_0, p_0, r_0, q_0, q_1, \ldots, r_1, q_{k+1}, \ldots, p_1, q_{m+1}, q_{n+1}, \ldots\}$. Here i_0 refers to the time the real information need (*RIN*) came to existence, p_0 is the time when the user perceived it, and decided what he wants to search for, r_0 is the time when he formulated a natural-language request, and q_0 is the time when that request turned into a query for a search engine. Proceeding further, q_1 is the time of the first re-formulation of the request into a different query, r_1 is the first re-formulation of the natural-language request, and p_1 is the first time when the user changed the perception of what he wants to find. A request change r_i is always followed by one or more attempts to re-formulate the query q_j, and for every change in user's *PIN* there is at least one attempt to formulate a new request.

4. **Components.** The final dimension of relevance in Mizarro's framework specifies the nature of relationship between the first and the second dimension. It can be *topical* relevance, *task* relevance, *context* relevance, or any combination of the three. *Topical* relevance is concerned with semantic similarity in the content of the two items. *Task* relevance specifies that the item or information contained in it is useful for the task the user is performing. *Context* includes anything that is not covered by the topic and the task. Mizarro's *context* is a kind of "miscellaneous" category that subsumes the notions of novelty, comprehensibility, search cost, and everything else that does not seem to fit elsewhere in his formulation.

Mizarro [89] argues that his framework provides a useful tool for classifying and comparing various definitions of relevance. For example, the simple definition we provided in section 2.1.1 corresponds to *topical* relevance of a *surrogate* to the *query* at time q_0. To contrast that, Vickery's [139, 140] notion of "user relevance" relates the *information* in a document to the real information need (*RIN*) with respect to *topic*, *task* and *context* together.

At this point it is helpful to pause and admit that most practical retrieval models assume a fairly conservative definition of relevance restricted to a small portion of Mizarro's space. Most operational systems are concerned exclusively with *topical* relevance of full-text *documents* to natural-language *requests*. While there is active research on modeling *task* relevance, or taking *context* into account, the majority of experimental studies address only topicality. Any kind of relevance with respect to perceived information need (*PIN*) is difficult to assess since there is only one user that knows what that *PIN* is. Furthermore, if we are dealing with *PIN*, we cannot cross-validate relevance judgments across different annotators. Modeling the real information need (*RIN*) is next to infeasible, since no one (not even the user) really knows what that is. Similarly, with very few exceptions (e.g. [78]) no one attempts to explicitly model the *information* contained in a given document – we simply do not have the frameworks that are simultaneously rich enough to represent arbitrary human discourse and robust enough to work on real-world data. On the other end of the spectrum, document *surrogates* and boolean *queries*, while quite popular in the past, are rarely used by the modern retrieval engines. Most current systems use full-text indexing, often with positional information and directly support natural-language requests.

2.2.1 Relevance in this thesis

For the scope of this thesis, we will concern ourselves with the popular view of relevance. In terms of Mizarro's classification, we will be constructing a formal model of relevance dealing with:

1. **Documents.** We will be operating on complete surface representations of information items (i.e. full text of documents, image bitmaps, segments of video, etc.)

2. **Requests.** We are concerned only with observable natural-language requests, for which we can obtain relevance judgments. However, our model will involve a notion similar to the real information need (**RIN**), which will play the role of a latent variable. We will use the words "query" and "request" interchangeably.

3. **Non-interactive.** We will not be modeling any evolution of user's information need. Our model explicitly accounts for the fact that an information need can be expressed in multiple forms, but we do not view these in the context of an interactive search session.

4. **Topicality.** We will be interested exclusively in *topical* relevance, i.e. the semantic correspondence between a given request and a given document. We will not be addressing issues of novelty, readability and suitability to a particular task.

2.3 Existing Models of Relevance

This thesis is certainly not the first endeavor to treat relevance in probabilistic terms. Some of the more prominent examples are the 2-Poisson indexing model developed independently by Bookstein and Swanson [15, 16] and Harter [52], the probabilistic retrieval model of Robertson and Sparck Jones [113], the probabilistic flavors [119] of Van Rijsbergen's logical model [136], the inference-network model developed by Turtle and Croft [131, 130], the language modeling approach pioneered by Ponte and Croft [102, 101] and further developed by others [87, 56], and the recent risk minimization framework of Zhai and Lafferty [66, 153]. While we cannot provide a comprehensive review of all probabilistic models, we will devote the remainder of this chapter to a brief description of those models that had a particularly strong influence on the development of our generative model.

2.3.1 The Probability Ranking Principle

At the foundation of just about all probabilistic models of IR lies an intuitive principle that a good search system should present the documents in order of their probability of being relevant to the user's request. It appears that this idea was first formally stated by Cooper in a private communication to Robertson, who published it in the following form [110]:

> The Probability Ranking Principle (PRP): If a reference retrieval system's response to each request is a ranking of the documents in the collections in order of decreasing probability of usefulness to the user who submitted the request, where the probabilities are estimated as accurately as possible on the basis of whatever data has been made available to the system for this purpose, then the overall effectiveness of the system to its users will be the best that is obtainable on the basis of that data.

Symbolically, if D is used to denote every observable property of a given document in a search task, and if R is a binary variable indicating whether or not that document is relevant to the search request, then optimal performance would result from ranking documents in the decreasing order of $P(R = 1|D)$. The words "optimal performance" refer both to informal user satisfaction (as long as redundancy is ignored) and to formal measures of evaluating IR performance. For example, it is not difficult to prove that using PRP to rank all documents in a given collection will lead to the greatest expected number of relevant hits in the top n ranks for every value of n (see [110]). Once that is established, it becomes obvious that PRP maximizes *recall* and *precision* at rank n[1], as well as any measures that are derived from recall and precision, such as the F-measure [133], R-precision and mean average precision (MAP). Robertson [110] and Van Rijsbergen [133] also demonstrate that the Probability Ranking Principle leads to minimal decision-theoretic loss associated with retrieving a set of n top-ranked documents.

The probability ranking principle, as stated by Robertson is quite broad – it does not restrict us to any particular type of relevance, and the document representation D can be potentially very rich, covering not just the topical content of the document. In fact D could include features determining readability of the document, its relation to the user's preferences or suitability for a particular task. Similarly, R may well refer to "pertinence" or any other complex notion of relevance. The only restriction imposed by the PRP is that relevance of a particular document be independent of any other document in the collection. Consequently, PRP cannot handle issues of novelty and redundancy, or cases where two documents are relevant when put

[1] *Recall* is defined as the number of relevant hits in the first n ranks, divided by the total number of relevant documents. *Precision* is the number of relevant hits in the first n ranks, divided by n.

together, but irrelevant when viewed individually. Robertson [110] also cites a curious counter-example (due to Cooper) regarding the optimality of the principle. The counter-example considers the case when we are dealing with two classes of users who happen to issue the same request but consider different documents to be relevant to it. In that case PRP will only be optimal for the larger of the two user classes.

While PRP itself is quite general, in practice most probabilistic models take a somewhat more narrow view of it. In most cases the relevance R is restricted to mean *topical* relevance of a full-text document to a natural-language request. Relevance is also assumed to be fixed *a priori* in a form of unobserved relevance judgments for every document in the collection. These judgments are not directly observable by the system, but it is assumed that they exist and that they do not change in the course of a search session.

In the remainder of this section we will take a detailed look at several prominent probabilistic models of information retrieval. All of these models are either directly based on the PRP, or can be closely related to the principle. The major distinction between the models lies in how the authors choose to estimate the probability of relevance $P(R = 1|D)$.

2.3.2 The Classical Probabilistic Model

We will first consider a probabilistic model of retrieval proposed by Robertson and Sparck Jones in [113]. The model was initially named the *Binary Independence Model*, reflecting the basic assumptions it made about occurrences of words in documents. However, since 1976 the model has been re-formulated a number of times to a degree where it can hardly be called "binary" and, as we shall argue later on, the term "independence" is also questionable. The model is also known as the *Okapi* model, the *City* model, or as the *classical probabilistic* model. A very detailed account of the recent state of the model is provided by the original authors in [127, 128]. What follows is our interpretation of the model. An attentive reader may find that our description is different in two ways from the way the model is presented by the authors [113, 127, 128]. First, we choose to describe the model in terms of probabilities, as opposed to log-likelihood weights. Second, we make explicit some of the assumptions that are never stated by the authors, particularly in section 2.3.2.2. Both steps are taken to make the description more compatible with subsequent material.

Robertson and Sparck Jones start the development of their model by transforming the probability ranking principle into the rank-equivalent likelihood ratio:

$$P(R=1|D) \propto \frac{P(R=1|D)}{P(R=0|D)} = \frac{P(D|R=1)P(R=1)}{P(D|R=0)P(R=0)} \propto \frac{P(D|R=1)}{P(D|R=0)} \tag{2.1}$$

Here R is a Bernoulli random variable indicating whether or not a given document is relevant and D is a random variable representing the contents of that document. We assume that D takes values in some finite space $\mathcal{D}$, and that P represents a joint probability measure over $\{0,1\} \times \mathcal{D}$. The first step of equation (2.1) comes from the fact that R is a binary variable and $\frac{P}{1-P}$ is a monotone (rank-preserving) transformation of P. The second step is a straightforward application of Bayes' rule. In the third step we observe that the factor $\frac{P(R=1)}{P(R=0)}$ is a constant independent of D, and thus does not affect the ranking of documents. In order to proceed further we need to specify the nature of the document space $\mathcal{D}$. The space has to be flexible enough to capture the semantic content of the documents, and yet simple enough to allow estimation from limited data. Robertson and Sparck Jones take $\mathcal{D}$ to be the space of all subsets of the vocabulary $\mathcal{V}$ of our collection. Equivalently, a document D is a binary occurrence vector, such that $D_v = 1$ if word number v is present in the document, otherwise $D_v = 0$. The document space $\mathcal{D}$ is then the space of all possible binary vectors over the vocabulary: $\mathcal{D} = \{0,1\}^{N_\mathcal{V}}$, and the entire probability space of the model in question is $\{0,1\} \times \{0,1\}^{N_\mathcal{V}}$ – the same finite space as the space of $(N_\mathcal{V} + 1)$ coin tosses. As the next step in developing their model, Robertson and Sparck Jones assume that binary random variables D_i are mutually independent given the value of R, allowing them to factor the probabilities $P(D|R = 1)$ and $P(D|R = 0)$ as follows:

$$\frac{P(D{=}\vec{d}|R{=}1)}{P(D{=}\vec{d}|R{=}0)} = \prod_{v \in \mathcal{V}} \frac{P(D_v{=}d_v|R{=}1)}{P(D_v{=}d_v|R{=}0)} = \left(\prod_{v \in D} \frac{p_v}{q_v}\right)\left(\prod_{v \notin D} \frac{1-p_v}{1-q_v}\right) \tag{2.2}$$

The first step in equation (2.2) comes from assuming independence between random variables D_v. We honor the common practice of using capital letters (e.g. D_v) to denote random variables and lowercase letters (e.g.

d_v) to refer to their observed values. The second step comes from the fact that D_v can only take values of 0 and 1, and using a shorthand $p_v = P(D_v{=}1|R{=}1)$ and $q_v = P(D_v{=}1|R{=}0)$. Also note the slight abuse of notation in the product subscripts: expression $v{\in}D$ really means $\{v{\in}\mathcal{V} : d_v{=}1\}$. As the final step Robertson and Sparck Jones desire that an empty document ($\vec{0}$) correspond to a natural zero in the log-space of their ranking formula. They achieve this by dividing equation (2.2) by $\frac{P(\vec{0}|R=1)}{P(\vec{0}|R=0)}$. Since that quantity is independent of any document, dividing by it will not affect document ranking and will yield the following final ranking criterion:

$$P(R{=}1|D{=}\vec{d}) \propto \frac{P(D{=}\vec{d}|R{=}1)}{P(D{=}\vec{d}|R{=}0)} / \frac{P(D{=}\vec{0}|R{=}1)}{P(D{=}\vec{0}|R{=}0)} = \prod_{v \in D} \frac{p_v(1-q_v)}{q_v(1-p_v)} \tag{2.3}$$

2.3.2.1 Parameter estimation with relevance information

Next comes the problem of estimation: in order to apply equation (2.3) to document retrieval we need to specify the quantities p_v and q_v, which reflect the probabilities of the word v occurring in a relevant and a non-relevant document respectively. Robertson and Sparck Jones start with the case where the relevance variable R is observable, that is for every document $D{=}\vec{d}$ in the collection they know whether it is relevant or not. If that is the case, a natural estimate of p_v is the proportion of relevant documents that contain word v, and similarly q_v is the proportion of non-relevant documents containing v. However, when R is fully observable, there is really no point in ranking: we could simply return the documents for which $R = 1$. A more realistic case is when R is partially observable, i.e. for some documents we know whether they are relevant or not, for others R is unknown. This is precisely the environment in Information Filtering, Topic Tracking or Relevance Feedback tasks. For that case Robertson and Sparck Jones adjust the relative frequencies of v in the relevant and non-relevant documents by adding a constant 0.5 to all counts:

$$p_v = \frac{N_{1,v} + 0.5}{N_1 + 0.5} \qquad q_v = \frac{N_{0,v} + 0.5}{N_0 + 0.5} \tag{2.4}$$

Here N_1 is the number of known relevant documents, $N_{1,v}$ of them contain the word v. Similarly N_0 and $N_{0,v}$ reflect the total number of known non-relevant documents and how many of them contain v. The constant 0.5 in equation (2.4) serves two purposes: first, it ensures that we never get zero probabilities for any word v, and second, it serves as a crude form of smoothing, reducing the variance of estimates over possible sets of feedback documents.

2.3.2.2 Parameter estimation without relevance information

Until this point development of the Okapi model was quite straightforward. Unfortunately, it was based on the relevance variable R being at least partially observable, and that is simply not the case in a typical retrieval environment. In a typical scenario the only thing we have is the user's request Q, usually expressed as a short sentence or a small set of keywords. All our probability estimates have to be based on Q and on the collection *en masse*, without knowing relevance and non-relevance of any documents. To complicate matters further, Q is not even present in the original definition of the model (eqs. 2.1-2.3), it becomes necessary only when we have no way of observing the relevance variable R. Faced with these difficulties, Robertson and Sparck Jones make the following assumptions:

1. $p_v{=}q_v$ if $v{\notin}Q$. When a word is not present in the query, it has an equal probability of occurring in the relevant and non-relevant documents. The effect of this assumption is that the product in equation (2.3) will only include words that occur both in the document and in the query, all other terms cancel out.

2. $p_v{=}0.5$ if $v{\in}Q$. If a word does occur in the query, it is equally likely to be present or absent in a relevant document. The assumption was originally proposed by Croft and Harper [34] and later re-formulated by Robertson and Walker [108]. The effect is that p_v and $(1-p_v)$ cancel out in equation (2.3), leaving only $\frac{1-q_v}{q_v}$ under the product.

3. $q_v{\propto}N_v/N$. Probability of a word occurring in a non-relevant document can be approximated by its relative frequency in the entire collection. Here N is the total number of documents in the collection, N_v

of them contain v. This approximation makes $\frac{1-q_v}{q_v}$ be proportional to the *inverse document frequency* (IDF) weight – a simple but devilishly effective heuristic introduced by Sparck Jones in [124].

Note that assumption (3) is quite reasonable: for a typical request only a small proportion of documents will be relevant, so collection-wide statistics are a good approximation to the non-relevant distribution. The same cannot be said for assumptions (1) and (2).

2.3.2.3 2-Poisson extension of the classical model

The original definition of the classical model deals exclusively with the binary representation of documents and queries, where a word is either present or not present in the document. However, empirical evidence suggests that the number of times a word is repeated within a document may be a strong indicator of relevance, and consequently the Okapi model was extended to include term frequency information. The first step in such extension is to expand the space $\mathcal{D}$ that is used to represent the documents. Previously, $\mathcal{D}$ was the set of all subsets of vocabulary $\{0,1\}^{N_\mathcal{V}}$. In order to handle frequencies, one can expand $\mathcal{D}$ to be $\{0,1,2\ldots\}^{N_\mathcal{V}}$. Now the ranking formula from equation (2.3) becomes:

$$P(R{=}1|D{=}\vec{d}) \propto \frac{P(D{=}\vec{d}|R{=}1)}{P(D{=}\vec{d}|R{=}0)} \Big/ \frac{P(D{=}\vec{0}|R{=}1)}{P(D{=}\vec{0}|R{=}0)} = \prod_{v \in D} \frac{p_v(d_v) q_v(0)}{q_v(d_v) p_v(0)} \tag{2.5}$$

Here d_v is the number of times word v was observed in the document. $p_v(d_v)$ is a shorthand for $P(D_v{=}d_v|R{=}1)$, the probability of seeing d_v of v in a relevant document, and $q_v(d_v)$ is the corresponding probability for the non-relevant document. Robertson and Sparck Jones [128] base their estimates of p_v and q_v on the 2-Poisson indexing model developed by Harter [52]. Harter's formalism revolves around a notion of *eliteness*, which was developed to model the behavior of a human indexer. Imagine a librarian who decides which keywords should be assigned to a given document. If he picks word v as a keyword for document d, then we say that d belongs to the *elite* class of v. Otherwise d belongs to the *non-elite* class. We would expect that documents in the elite class of v are likely to contain many repetitions of v, while in the non-elite class v would primarily occur by chance. Harter assumed that frequency of v in both classes follows a Poisson distribution, but that the mean is higher in the elite class. Under this assumption, the frequency of v in the collection *en masse* would follow a mixture of two Poissons:

$$P(D_v{=}d_v) = P(E{=}1)\frac{e^{-\mu_{1,v}}\mu_{1,v}^{d_v}}{d_v!} + P(E{=}0)\frac{e^{-\mu_{0,v}}\mu_{0,v}^{d_v}}{d_v!} \tag{2.6}$$

Here E is a binary variable specifying whether D is in the elite set of v, $\mu_{1,v}$ is the mean frequency of v in the elite documents, and $\mu_{0,v}$ is the same for the non-elite set. Since we don't know which documents are elite for a given word, we need some way to estimate three parameters: $\mu_{1,v}$, $\mu_{0,v}$ and $P(E{=}1)$. Harter's solution was to fit equation (2.6) to the empirical distribution of v in the collection using the method of moments. But eliteness is not quite the same as thing as relevance, since eliteness is defined for single words and cannot be trivially generalized to multi-word requests. In order to fit Harter's model into the Okapi model the authors had to make some adjustments. Robertson and Walker [114] proposed to condition eliteness on R, and assumed that once we know eliteness, the frequency of v in a document is independent of relevance:

$$p_v(d_v) = P(E{=}1|R{=}1)\frac{e^{-\mu_{1,v}}\mu_{1,v}^{d_v}}{d_v!} + P(E{=}0|R{=}1)\frac{e^{-\mu_{0,v}}\mu_{0,v}^{d_v}}{d_v!} \tag{2.7}$$

$$q_v(d_v) = P(E{=}1|R{=}0)\frac{e^{-\mu_{1,v}}\mu_{1,v}^{d_v}}{d_v!} + P(E{=}0|R{=}0)\frac{e^{-\mu_{0,v}}\mu_{0,v}^{d_v}}{d_v!} \tag{2.8}$$

Substituting equations (2.7,2.8) back into the ranking formula (2.5), leads to a rather messy expression with a total of 4 parameters that need to be estimated for every word v: the mean frequencies in the elite and non-elite sets ($\mu_{1,v}$ and $\mu_{0,v}$), and the probability of eliteness given relevance or non-relevance ($P(E{=}1|R{=}1)$ and $P(E{=}1|R{=}0)$). This presents a rather daunting task in the absence of any relevance observations, leading

the authors to abandon formal derivation and resort to a heuristic. They hypothesize that equations (2.7,2.8) might lead to the following term under the product in equation (2.5):

$$\frac{p_v(d_v)q_v(0)}{q_v(d_v)p_v(0)} \approx \exp\left\{\frac{d_v \cdot (1+k)}{d_v + k \cdot \left((1-b) + b\frac{n_d}{n_{avg}}\right)} \times \log\frac{N}{N_v}\right\} \tag{2.9}$$

The quantity under the exponent in equation 2.9 represents the well-known and successful *BM25* weighting formula. As before, d_v is the number of times v occurred in the document, n_d is the length of the document, n_{avg} is the average document length in the collection, N is the number of documents in the collection and N_v is the number of documents containing v. k and b represent constants that can be tuned to optimize performance of the model on the task at hand. At this point it is appropriate to stress the fact that equation (2.9) is not a derived result and does not result from any meaningful assumptions about the constituents of equations (2.7,2.8). *BM25* is a work of art, carefully engineered to combine the variables that were empirically found to influence retrieval performance: term frequency, document length and inverse document frequency. It is simple, flexible and very effective on a number of tasks (see [127, 128]). Unfortunately, it has no interpretation within the probabilistic model.

2.3.2.4 Modeling dependence in the classical model

The assumption of word independence in the classical model is a favorite target of linguistically sophisticated critics and aspiring graduate students alike. No other aspect of the formalism has drawn so much criticism and so many failed endeavors to improve the model[2]. Recall that the assumption states that individual words D_i in the document are mutually independent given the relevance variable R. The assumption is formalized in equation (2.2) for binary document representation and in equation (2.5) for the non-binary case. The assumption is intuitively wrong – we know that words in a language are not independent of each other: supposing that presence of the word "politics" tells us nothing about occurrence of "Washington" is clearly absurd. The popular perception is that the assumption of independence is a necessary evil, it is tolerated simply because without it we would have to estimate joint probabilities for vectors involving half a million random variables each (typical vocabulary size), and that is clearly intractable. Another popular perception is that there must be a way to partially model these dependencies, bringing the model closer to reality, and surely improving the retrieval performance.

One of the first attempts to relax the independence assumption is due to Van Rijsbergen [132]. The idea is to allow pairwise dependencies between words, such that for every word v there exists a *parent* word $\pi(v)$ which influences presence or absence of v. There is also a *root* word v_0 which has no parent. Dependencies form a spanning tree over the entire vocabulary, the structure of that tree can be induced automatically from a corpus by maximizing some objective function. Van Rijsbergen suggested using the aggregate mutual information over the branches: $\sum_v I(v, \pi(v))$, other measures may work equally well. Once the structure $\pi(\cdot)$ is determined, we can replace the probabilities $P(D_v{=}d_v|R)$ in equations (2.2) and (2.5) with their conditional counterparts $P(D_v{=}d_v|D_{\pi(v)}{=}d_{\pi(v)}, R)$. After that we re-arrange the indices v in the products to descend down the tree, and voila, we have a way to model relevance without assuming mutual independence.

Unfortunately, empirical evaluations [51, 50] of the new model suggest that by and large it performs no better than the original. When improvements were observed they were mostly attributed to *expanding* the query with additional words, rather than more accurate modeling of probabilities. Disappointing performance of complex models is often blamed on combinatorial explosion of the number of parameters. However, in Van Rijsergen's model the total number of parameters is only twice that of the original formulation: we replace p_v in equation (2.2) with $p_{v,0}$ and $p_{v,1}$, reflecting absence and presence of $\pi(v)$; the same is done for q_v. Neither can we blame performance on the particular choices made in [132] – during the two decades that passed, Van Rijsbergen's idea has been re-discovered and re-formulated by various researchers in wildly different ways [26, 27, 64, 77, 106, 133, 151]. In most cases the results are disappointing: consistent improvement is only reported for very selective heuristics (phrases, query expansion), which cannot be treated as formal models of word dependence. The pattern holds both when relevance is not observable (ad-hoc retrieval) and

[2] It is my personal observation that almost every mathematically inclined graduate student in Information Retrieval attempts to formulate some sort of a non-independent model of IR within the first two or three years of his studies. The vast majority of these attempts yield no improvements and remain unpublished.

when there are a lot of relevant examples (text classification). In the latter case even phrases are of minimal value.

2.3.2.5 Why dependency models fail

It is natural to wonder why this is the case – the classical model contains an obviously incorrect assumption about the language, and yet most attempts to relax that assumption produce no consistent improvements whatsoever. In this section we will present a possible explanation. We are going to argue that the classical Binary Independence Model really *does not* assume word independence, and consequently that there is no benefit in trying to relax the non-existent assumption. Our explanation is an extension of a very important but almost universally ignored argument made by Cooper in [29]. Cooper argues that in order to arrive at equation (2.2), we only need to assume *linked dependence* between words D_i, and that assumption is substantially weaker than independence. Cooper's argument is as follows. Consider the case of a two-word vocabulary $\mathcal{V}=\{a, b\}$, and suppose we do not assume independence, so $P_1(D_a, D_b)$ is their joint distribution in the relevant class, $P_0(D_a, D_b)$ is the same for the non-relevant class. Now for a given document $D=\vec{d}$, define $k_1=\frac{P_1(d_a,d_b)}{P_1(d_a)P_1(d_b)}$ and $k_0=\frac{P_0(d_a,d_b)}{P_0(d_a)P_0(d_b)}$. By definition, k_1 is a measure of dependence between events $D_a=d_a$ and $D_b=d_b$ in the relevant class; it tells as how wrong we would be if we assumed D_a to be independent of D_b. If $k_1>1$, the events d_a and d_b are positively correlated in the relevant class, $k_1<1$ means they are negatively correlated. k_0 plays the same role for the non-relevant class. Without assuming independence, the posterior odds of relevance (equation 2.1) takes the form:

$$P(R=1|D=\vec{d}) \propto \frac{P(D=\vec{d}|R=1)}{P(D=\vec{d}|R=0)} = \frac{P_1(d_a, d_b)}{P_0(d_a, d_b)} = \frac{k_1 P_1(d_a) P_1(d_b)}{k_0 P_0(d_a) P_0(d_b)} \tag{2.10}$$

When Robertson and Sparck Jones [113] assume that D_a and D_b are independent, they in effect assume that $k_1=1$ and $k_0=1$. But Cooper [29] correctly points out that to justify equation (2.2) we only need to assume $k_1=k_0$, which is much less restrictive: k_1 and k_2 can equal any number, not just 1. This is Cooper's *linked dependence* assumption, it demonstrates that the classical model actually allows for any degree of dependence between words a an b, as long as that dependence is exactly the same in the relevant and non-relevant classes.

Cooper's assumption is certainly more reasonable than mutual independence, but it has its limitations. For example, if the user's request happens to deal with compound concepts, such as "machine translation", it would be disastrous to assume the same degree of dependence for these two words in the relevant and non-relevant documents. Additionally, linked dependence presented by Cooper, becomes more and more restrictive as we consider larger vocabularies and deal with factors k_1 and k_0 of the form $\frac{P(d_1 \ldots d_n)}{P(d_1) \ldots P(d_n)}$. However, we would like to argue that Cooper's argument can be taken one step further, yielding an even weaker **proportional interdependence** assumption that can withstand the counterexample given above. Let $\mathcal{V}$ be a general vocabulary. We will restrict our discussion to the simple case where only first-order dependencies exist between the words: a word v may only depend on one other word, as in Van Rijsbergen's model [132]. We will go a step further and allow each word v to have potentially different parents $\pi_1(v)$ and $\pi_0(v)$ in the relevant and non-relevant dependence trees. We know that under a first-order model, the joint distribution $P(D=\vec{d}|R=r)$ decomposes into a product of conditional probabilities $P(D_v=d_v|D_{\pi_r(v)}=d_{\pi_r(v)}, R=r)$, one for each word v in the vocabulary. Inspired by Cooper, we define the factor $k_{v,r}$ to be the ratio of the conditional probability to the unconditional one: $k_{v,r}=\frac{P(D_v=d_v|D_{\pi_r(v)}=d_{\pi_r(v)}, R=r)}{P(D_v=d_v|R=r)}$. Now the version of equation (2.1) appropriate for a first-order dependence model will take the following form:

$$P(R=1|D=\vec{d}) \propto \prod_{v \in \mathcal{V}} \frac{P(D_v=d_v|D_{\pi_1(v)}=d_{\pi_1(v)}, R=1)}{P(D_v=d_v|D_{\pi_0(v)}=d_{\pi_0(v)}, R=0)} = \prod_{v \in \mathcal{V}} \frac{P(D_v=d_v|R=1)}{P(D_v=d_v|R=0)} \cdot \frac{k_{v,1}}{k_{v,0}} \tag{2.11}$$

Equation (2.11) makes it very clear that the first-order model is rank-equivalent to the independence model if and only if $\prod_v \frac{k_{v,1}}{k_{v,0}}$ is a constant independent of $\vec{d}$. An equivalent statement is that cumulative pairwise

mutual information between the presence of a randomly picked word and the presence of its parent differs by a constant k in the relevant and non-relevant classes:

$$\sum_{v \in \mathcal{V}} \log \frac{P_1(d_v, d_{\pi_1(v)})}{P_1(d_v)P_1(d_{\pi_1(v)})} - \sum_{v \in \mathcal{V}} \log \frac{P_0(d_v, d_{\pi_0(v)})}{P_0(d_v)P_0(d_{\pi_0(v)})} = k : k \perp \vec{d} \tag{2.12}$$

Informally, equation (2.12) means that *on average*, words in a given document have about as much interdependence in the relevant class (P_1) as they do in the non-relevant class (P_0). The key phrase here is "on average": equation (2.12) does not require that any two words be equally dependent under R=0 and R=1 (that is precisely Cooper's linked dependence). Instead, equation (2.12) allows some words to be strongly dependent only in the relevant class (e.g., "machine" and "translation"), as long as on average they are balanced out by some dependencies that are stronger in the non-relevant class. They don't even have to balance out exactly (k=0), the only requirement is that whatever disbalance exists be constant across all documents.

We view the above as a strong result: the independence model is equivalent to any first-order dependence model under a very weak **proportional interdependence** assumption, that we personally believe holds in most situations.[3] Indeed, we see no reason to believe that an arbitrary set of documents in the collection (the relevant set for some request) will exhibit a stronger cumulative dependence over all words than will the complement of that set. The meaning of this result is that any attempt to replace independence with first-order dependence is very likely to produce no improvements, other than by accident. We also would like to point out that this result may not be limited to first-order dependencies. One could define factors $k_{v,0}$ and $k_{v,1}$ for higher-order models where word probabilities are conditioned on *neighborhoods* $\eta(v)$ instead of of *parents* $\pi(v)$. Admittedly, the argument becomes somewhat more elaborate; we have not worked out the details. As a conclusion to this section, we would like to stress the following:

> Contrary to popular belief, word independence is not a necessary assumption in the classical probabilistic model of IR. A necessary and sufficient condition is proportional interdependence, which we believe holds in most retrieval settings. If there is anything wrong with the classical model, it is not independence but the assumptions made in the estimation process (see sections 2.3.2.2 and 2.3.2.3).

2.3.3 The Language Modeling Framework

We will now turn our attention to a very different approach to relevance – one based on statistical models of natural language. Statistical language modeling is a mature field with a wide range of successful applications, such as discourse generation, automatic speech recognition and statistical machine translation. However, using language modeling in the field of Information Retrieval is a relatively novel development. The approach was proposed by Ponte and Croft [102, 101] in 1998, and in the short time since then it has attracted a tremendous level of interest and a growing number of publications each year. In this section we will outline the original language modeling approach to IR [102] and briefly mention some of the more prominent extensions.

One of the main motivations Ponte and Croft had for developing the language modeling approach was to get away from the heuristics that came to dominate the probabilistic model of Robertson and Sparck Jones [113]. Recall that heuristics in the classical model arise when we are given no examples to estimate the probabilities p_v associated with relevant documents. Ponte and Croft's solution to this problem was quite radical – it was to remove the explicit relevance variable R, and construct a probabilistic model around the document and the user's query. The authors hypothesized that for every document D=d, there exists an underlying language model M_d. Now, if the query Q=q looks like it might be a random sample from M_d, we might have a reason to believe that d is relevant to q. Informally, we can think of M_d as a crude model reflecting the state of mind of the author who created document d. If the same state of mind is likely to produce the query q, then it is likely that q is topically related to d, hence d would be topically relevant.

[3] If desired, one could certainly test the empirical validity of the proportional interedependence assumption for a given collection. The test would proceed as follows. **(1)** partition the collection into relevant and non-relevant sets using complete relevance judgments. **(2)** estimate the dependency structures $\pi_1()$ and $\pi_0()$ for the relevant and non-relevant classes (e.g., using Van Rijsbergen's method). **(3)** construct maximum-likelihood estimates for conditional distributions $P_r(v|\pi_r(v)) : r \in \{0, 1\}$. **(4)** compute the value k in equation (2.12) for each document d in the collection. **(5)** plot these values against the relevance of d (0 or 1). If there is a clear trend, the proportional interdependence assumption does not hold.

The effect of Ponte and Croft's argument is that they could replace the probability of relevance $P(R{=}1|D{=}d)$ with the probability of observing the query from the language model of the document $P(Q{=}q|M_d)$. This is a crucial step: it allows the authors to avoid the uncertainty associated with the unobserved relevance variable R. Indeed, Q is observable, and there exists a substantial body of statistical literature to help us in estimating M_d from the observed document d. Retrieval in Ponte and Croft's model can be decomposed into two steps. First, we have to use the observation d to construct our estimate of the underlying document language model M_d. Second, we can compute the probability $P(Q{=}q|M_d)$ of observing q as a random sample from M_d, and rank all documents in the decreasing order of that probability.

2.3.3.1 Multiple-Bernoulli language models

Ponte and Croft represent queries in the same space that was used by Robertson and Sparck Jones for their Binary Independence Model. If $\mathcal{V}$ is a vocabulary of $N_\mathcal{V}$ words, the query space $\mathcal{Q}$ is the set of all subsets of vocabulary ($\{0,1\}^{N_\mathcal{V}}$). The query Q is a vector of $N_\mathcal{V}$ binary variables Q_v, one for each word v in the vocabulary. The components Q_i are assumed to be mutually independent conditioned on the language model M_d. The language model itself is a vector of $N_\mathcal{V}$ probabilities $p_{d,v}$, one for each word v. The probability of observing a query $Q{=}\vec{q}$ from a given model $M_d{=}\vec{p}_d$ is:

$$P(Q{=}\vec{q}|M_d{=}\vec{p}_d) \;=\; \prod_{v\in\mathcal{V}} P(Q_v{=}q_v|M_d{=}\vec{p}_d) \;=\; \prod_{v\in Q} p_{d,v} \times \prod_{v\notin Q}(1-p_{d,v}) \tag{2.13}$$

Here again, $v{\in}Q$ is a shorthand for $\{v{\in}\mathcal{V} : q_v{=}1\}$, and likewise for the complement set. Ponte and Croft propose the following way to compute $\vec{p}_d$ from the document d:

$$p_{v,d} = \begin{cases} \left(\frac{d_v}{|d|}\right)^{(1-r)} \left(\frac{1}{N_v}\sum_{d'} \frac{d'_v}{|d'|}\right)^{r} & \text{if } d_v > 0 \\ \left(\sum_{d'} d'_v\right) / \left(\sum_{d'} |d'|\right) & \text{otherwise} \end{cases} \tag{2.14}$$

Here d_v is the number of times word v occurs in document d, $|d|$ denotes the length of document d, N_v is the number of documents containing v and the summations go over every document d' in the collection. If a word v does not occur in the document, Ponte and Croft use its relative frequency in the entire collection. If a word does occur, the estimate is a weighted geometric average between the relative frequency in the document and the average relative frequency over all documents containing v. The weight is given by the parameter r, which according to the authors plays the role of *Bayesian risk*.

A probabilist will undoubtedly notice an inconsistency between equations (2.13) and (2.14). The former represents a *multiple-Bernoulli* distribution over the binary event space, but the probability estimates in equation (2.14) are based on non-binary frequencies d_v and would naturally arise if we assumed that d was a random sample from a *multinomial* distribution. Ponte and Croft never address the issue, but elsewhere [85] we show that the model can be made consistent by assuming that d is a *set* of $|d|$ binary vectors, each drawn from M_d.

2.3.3.2 Multinomial language models

As we mentioned above, term frequencies are somewhat unnatural in Ponte and Croft's model. The model is explicitly geared to capture the presence or absence of words, and does not recognize the fact that words can be repeated in the request. This is perfectly reasonable for short 2-3 word queries that are typical of web searches, but it is not a good assumption for the general retrieval setting. In order to take account of frequencies researchers have had to assume a different event space. Virtually every publication concerning language modeling in IR [122, 123, 87, 11, 148, 58, 56, 154, 155] presumes the following representation, though it is rarely stated in formal terms. Assume $\mathcal{V}$ is a vocabulary of $N_\mathcal{V}$ distinct words. Both documents and queries are viewed as strings (sequences) over $\mathcal{V}$. A document D of length n is a sequence of n random variables D_i, each taking values in the vocabulary $\mathcal{V}$. The query Q has the same representation: $Q{=}\{N{\in}\mathbb{N}, Q_i{\in}\mathcal{V} : i{=}1\ldots N\}$. The probability space $\mathcal{D}$ for both documents and queries is the space of all possible sequences of words: $\mathcal{D} = \mathbb{N}\times\cup_{n=1}^{\infty}\mathcal{V}^n$. Note: most authors omit sequence length N from the representation. We make it explicit to define a single probability measure for strings of any length. Individual words Q_i in the sequence are assumed to be independent of N, independent of each other and identically

distributed according to the language model M_d. M_d now plays the role of a discrete distribution over the vocabulary; its values are vectors of $N_\mathcal{V}$ probabilities, one for each word v: $M_d{=}\{\vec{p}_d{\in}[0,1]^{N_\mathcal{V}} : |\vec{p}_d|{=}1\}$. The probability mass assigned to string Q is:

$$P(Q{=}q|M_d{=}\vec{p}_d) \quad = \quad P(N{=}n)\prod_{i=1}^{n} P(Q_i{=}q_i|M_d{=}\vec{p}_d) \quad = \quad \pi_n \prod_{i=1}^{n} p_{d,q_i} \tag{2.15}$$

Here $P(N{=}n) = \pi_n$ is some discrete distribution over string lengths; it is independent of everything else and is usually assumed to be uniform until some maximum length M and zero beyond that. A common way to estimate the language model is to assume that the document d represents a random sample from M_d and use relative frequencies of words in d as a maximum likelihood estimate $\vec{p}_d$. However, maximum likelihood estimation will naturally lead to zeros in the estimate, so some form of smoothing is required. From the large pool of available smoothing techniques [23, 154], most authors pick some form of linear interpolation between the maximum likelihood estimate and the "background" frequency of a word computed over the whole collection:

$$p_{d,v} = \lambda_d \frac{n_{d,v}}{n_d} + (1-\lambda_d)\frac{n_{c,v}}{n_c} \tag{2.16}$$

Here $n_{d,v}$ refers to the number of times the word v occurs in document d, n_d is the length of d, $n_{c,v}$ is the frequency of v in the entire collection and n_c is the total number of words in the collection. Parameter λ_d is used to control the degree of variance in the estimator. Lafferty and Zhai [66] show that equation (2.16) is a natural Bayesian estimate that follows from assuming a Dirichlet prior with parameters proportional to $\frac{n_{c,v}}{n_c}$ over the simplex of all possible language models.

2.3.3.3 A note on multiple-Bernoulli vs. multinomial

As we already mentioned, the original language modeling framework proposed by Ponte and Croft [102] is defined over a binary event space, and does not allow for word repetitions. The multinomial approach described in section 2.3.3.2 does allow word frequencies, but that is not the only difference between the two frameworks. A much more important, and commonly overlooked difference is that the two approaches have very different and incompatible event spaces. The random variable Q means two completely different things in equations (2.13) and (2.15), and the meanings cannot be used interchangeably, or mixed together as was done in several publications. In Ponte and Croft's model, Q is a *vector* of binary variables Q_v. Each Q_v represents a *word* in the vocabulary, possible values of Q_v are 0 and 1 (absent and present). In the multinomial framework, Q is a sequence of variables Q_i, each Q_i represents a *position* in the query, and the values of Q_i are words. Note that the latter representation is absolutely not a requirement if we just want to model counts. We could have simply extended the range of Q_v in the Ponte and Croft model to include counts, as was done in the 2-Poisson extension of the Binary Independence Model. Doing this would give us a vector representation that is very different from the sequence representation described in section 2.3.3.2. To be specific, in the vector representation we have half a million random variables, each with two possible values, in the sequence we effectively have only one variable (since Q_i are i.i.d.), but that variable can take half a million possible values.

Which of these representations is more suitable for Information Retrieval is a very interesting open question. Our feeling is that vector representation might be more natural. It allows us to estimate a separate distribution for every vocabulary word, makes no a-priori assumptions about word counts and allows us to explicitly model dependencies between different vocabulary words (though in light of section 2.3.2.5 we might wonder if dependencies are of any use at all). On the other hand, the sequence representation makes it more natural to model word proximity, phrases and other surface characteristics of text. In practice, the question of representation is all but settled – nearly every publication assumes sequences rather than vectors. This choice is largely a matter of consistency – in the fields adjacent to IR, such as Speech Recognition (ASR), Machine Translation (MT) and Natural Language Processing (NLP), language modeling always concerns sequences. In addition, a large body of language modeling publications in these fields serves as a gold-mine of estimation techniques that can be applied in IR – anything from n-gram and cache models in ASR, to translation models in MT, to grammars in NLP. For the remainder of this thesis, when we speak of language models we will refer to sequence models, as defined in section 2.3.3.2.

2.3.3.4 Independence in the language modeling framework

Just like the classical probabilistic model, the language modeling approach relies on making a very strong assumption of independence between individual words. However, the meaning of this assumption is quite different in the two frameworks. In the classical model, independence means that presence of "politics" does not affect presence of "Washington". In the language modeling framework, the independence assumption means that the identity of the n'th word in the sequence does not depend on any preceding or following words. The second assumption implies the first: "Washington" in position n is still not affected by "politics" in position m. However the converse is not true: in addition to the above, the language modeling framework assumes that "politics" in position m does not affect "politics" in position n, so in a sense a word is independent of itself. This is certainly not the case in the classical model, although assuming a Poisson distribution in section 2.3.2.3 is essentially equivalent.

Given that the assumption of independence is even stronger in LM than in the classical model, it should come as no surprise that several researchers attempted to relax the assumption. One of the first attempts is due to Song and Croft [122, 123]. In that work each query word Q_i was conditioned on the immediately preceding Q_{i-1}, forming a first-order Markov Chain. The same assumption was made about the documents, and consequently the language model M_d takes the form of a bigram model. The parameters were estimated using bigram frequencies in the document with back-off to the unigram and to collection-wide counts. Note that in this case one does experience a combinatorial explosion: the number of parameters is squared, not just doubled as in Van Rijsbergen's dependence model [132]. As might be expected, the new model did not yield consistent improvements over the original formulation. A similar model was proposed by Miller and colleagues [86, 87], but bigram performance was never evaluated. A very different formalism was recently attempted by Nallapati [91, 92], who tried to combine Van Rijsbergen's dependence model [132] with the multinomial model in section (2.3.3.2). The results were inconsistent. We are also aware of several unpublished studies (including our own), where different attempts to introduce dependencies between random variables Q_i or D_i did not lead to any improvements over the unigram model.

Faced with the poor performance of dependency models, can we repeat the argument made in section 2.3.2.5 and show that document ranking is not affected by the assumption of independence? Our intuition is that we cannot: this time we are dealing with a distinct probability distribution M_d for each document d in the collection. Furthermore, interdependencies accumulate only over the query words, not over the entire vocabulary as was the case with the classical model. We see no reason to believe that aggregate dependence among the query words will not heavily depend on M_d. However, we will provide an informal argument for why modeling dependence does not seem to help in IR, whereas it is absolutely essential in other fields that use language models (ASR, MT, NLP). The primary use of language models in fields other than IR is to ensure surface consistency, or well-formedness of strings of words. As an example, consider a speech recognition system, which typically consists of an acoustic model and a language model. To an acoustic model the utterance "I see defeat" may appear no different from a nonsensical "icy the feet". But any decent bigram model would favor the first string as more consistent. Similarly, in NLP a discourse generation system may use a grammar of English to translate a template-based representation of some action into a well-formed sentence. In these fields it is absolutely necessary to worry about the surface form of strings because the goal is to generate *new* strings of text in response to some input. If the system generates gibberish, it is useless. Information retrieval is different in the sense that it deals with *existing* strings of text, which are already well-formed. When we directly adopt an n-gram model from speech recognition or a maximum-entropy model from MT, we are in effect adopting a proven solution for a problem that we do not face. At the same time, the added complexity of the new model will likely translate to less reliable parameter estimates than the ones we had with the simpler model.

The above argument should be taken with a grain of salt. We are not suggesting that it is impossible to improve the dependency structure of the language modeling approach. We only claim that no improvement should be expected from *direct* models of dependence – models where random variables Q_i are *directly* conditioned on some $Q_{j\neq i}$. That, however, does not mean that no improvement will result from models that capture dependencies *indirectly*, perhaps via a hidden variable. For example, Berger and Lafferty [11] proposed to model information retrieval as statistical translation of a document into the query. In the simplest instantiation of their approach (model 1), latent words T_i are randomly sampled from the document model M_d, and then probabilistically *translated* into query words Q_i according to some distribution $P(Q_i|T_i)$.

Under the translation model, the document ranking criterion (equation 2.15) takes the following form:

$$P(Q{=}q|M_d{=}\vec{p}_d) \;\propto\; \prod_{i=1}^{n} \sum_{v \in \mathcal{V}} P(Q_i{=}q_i|T_i{=}v) P(T_i{=}v|M_d{=}\vec{p}_d) \;=\; \prod_{i=1}^{n} \sum_{v \in \mathcal{V}} t_{v,q_i} p_{d,v} \tag{2.17}$$

In the translation model there is no direct dependency between the query words Q_i or the document words D_i. Instead, the translation matrix $t_{v,q}$ provides a useful way to handle dependencies between *identities* of the words in the document and in the query. For instance, the translation model would correctly model dependency between the word "cat" in a document and the word "feline" in the query, if the translation matrix $t_{v,q}$ was estimated to reflect synonymy. The translation model was originally proposed as a general-purpose model of IR, but it found its greatest application in the field of cross-language retrieval, where documents in one language are queried using some other language. Two other prominent examples of the indirect approach to word dependencies are the latent aspect model of Hoffman [58], and the Markov-chain model proposed by Lafferty and Zhai [66].

2.3.4 Contrasting the Classical Model and Language Models.

In the previous section we described two probabilistic frameworks for modeling relevance in information retrieval. One grew out of the Binary Independence Model, proposed by Robertson and Sparck Jones [113]; the other represents various developments of the language-modeling approach pioneered by Ponte and Croft [102]. This section will be devoted to looking at the benefits and drawbacks of the two frameworks and will serve as a bridge leading into the development of our own generative model of relevance.

Despite their fundamental differences, there are a number of similarities between the classical model and language models. Both approaches started with a focus on binary presence or absence of words, and, curiously, used exactly the same event space. Both rapidly evolved from binary occurrence to modeling word frequencies – one via a Poisson distribution over the counts, the other, implicitly, by adopting a multinomial distribution over the vocabulary. Both frameworks appear to make a very strong assumption of independence, though we have argued that the meaning of *independence* is quite different in the two models. In the former, independence concerns word *identities* (presence of "politics" unaffected by "Washington"); in the latter word *positions* are assumed independent (first query/document word does not affect second word). Since both assumptions appear obviously incorrect, a lot of effort went into improving performance by modeling dependencies. Unfortunately, explicit models of dependence did not lead to consistent performance improvements in either framework. In an attempt to understand this curious effect, we provided two different arguments for why dependency models do not help in the classical framework and the language modeling framework. For the classical case we extended an argument initiated by Cooper [29] and showed that the model really does not assume independence, it is based on a much weaker assumption of proportional interdependence (see section 2.3.2.5 for details). For the language modeling framework we informally argued that explicit models of dependence will capture nothing but the surface form (well-formedness) of text, which has little to do with the topical content.

2.3.4.1 Relevance in the two frameworks

The point where language models become very different from the classical models is on issue of relevance. The classical probabilistic model is centered around relevance: Robertson and Sparck Jones [113] start the development of their model directly from the probability ranking principle and proceed formally as far as they can (as long as relevance is observable). To contrast that, in the language modeling approach there is no explicit concept of relevance. Ponte and Croft [102] replace it with a simple generative formalism: probability of relevance $P(R|D)$ is assumed to be proportional to the probability of randomly drawing the query Q from the document model M_d. There is a clear benefit to this assumption: since both the query and the document are always fully observable, the model does not have to deal with the ambiguous concept of relevance. Ponte and Croft effectively turned a retrieval problem into an estimation problem. Instead of trying to model relevance we look for the best way to estimate the language model M_d for a given document d. This estimation step can be carried out in a systematic fashion, without resorting to heuristics that become necessary in the classical model.

However, absence of relevance raises the question of what to do in the rare cases when we do have relevant examples. To elaborate, suppose we have a small set of documents that are known to be relevant to a given query. How could we make use of this information to improve ranking of subsequent documents? The process is very straightforward in the classical probabilistic model – we simply use relative frequencies of words to get better probability estimates; the exact formulae are given in section 2.3.2.1. This can be done with both positive and negative (non-relevant) examples. With the language modeling framework incorporating relevance judgments is not nearly as clear. We cannot update probability estimates because there is no distribution associated with the relevant class. Updating the document model M_d makes no sense, since examples are relevant to the query, not to the document. Ponte [101] suggests that the only thing we can do is re-formulate the query, expanding it with words selected from relevant examples according to a heuristic weighting formula. Interestingly, language models allow for a very different kind of feedback that cannot be handled within the classical model. If for a given document d we have examples of relevant queries (queries for which d was judged relevant), we can certainly make use of those queries in adjusting the language model M_d. This form of feedback has recently been studied in [93].

Formal absence of relevance from the language modeling approach has also led to continued criticism of the framework [125, 107, 126]. Quoting from [126]: *"a retrieval model that does not mention relevance appears paradoxical"*. Responding to this criticism, Lafferty and Zhai [67] claim that the language modeling approach really *does* include a concept of relevance, albeit implicitly. To support their claim, Lafferty and Zhai argue that from a high-level viewpoint both frameworks operate with three random variables: the query Q, the document D and the relevance variable R. Both frameworks attempt to approximate the posterior distribution over R, but the factorization of dependencies is done in different ways. In the classical probabilistic model, D is factored out first and conditioned on R and Q, leading to the familiar development:

$$\frac{P(R{=}1|D,Q)}{P(R{=}0|D,Q)} \;=\; \frac{P(D|R{=}1,Q)}{P(D|R{=}0,Q)} \cdot \frac{P(R{=}1,Q)}{P(R{=}0,Q)} \;\propto\; \frac{P(D|R{=}1,Q)}{P(D|R{=}0,Q)} \tag{2.18}$$

The leftmost portion of Figure 2.1 shows a graphical diagram of the dependencies implied by equation (2.18). Following convention, we use shaded circles to represent observed variables D and Q. A corresponding dependency diagram for the language modeling approach is shown in the middle of Figure 2.1. We use dashed lines to indicate that R is introduced somewhat artificially. The diagram results from factoring Q conditioned on R and D as follows:

$$\frac{P(R{=}1|D,Q)}{P(R{=}0|D,Q)} = \frac{P(Q|R{=}1,D)}{P(Q|R{=}0,D)} \cdot \frac{P(R{=}1,D)}{P(R{=}0,D)} \propto P(Q|R{=}1,D) \cdot \frac{P(R{=}1,D)}{P(R{=}0,D)} \tag{2.19}$$

In order to justify the second step above, Lafferty and Zhai have to assume that Q is independent of D if $R{=}0$, which means that the denominator of the first ratio does not affect the ranking and can be omitted. However, equation (2.19) still includes the ratio of joint probabilities over R and D, which is not present in the language modeling approach. To get rid of it, Lafferty and Zhai proposed to make R independent of D. This would leave $P(Q|R{=}1,D)$ as the only term that affects the ranking of documents, thus explaining the language modeling framework.

As a final note, we would like to suggest that there is a third way of factoring the joint distribution over R, D and Q. We could assume that the query Q and the document D are conditioned on the relevance variable R, and that Q and D are independent given R. This factorization is shown in the rightmost diagram in Figure 2.1, it forms the foundation for the model proposed in this thesis and will be discussed in great detail in the following chapter.

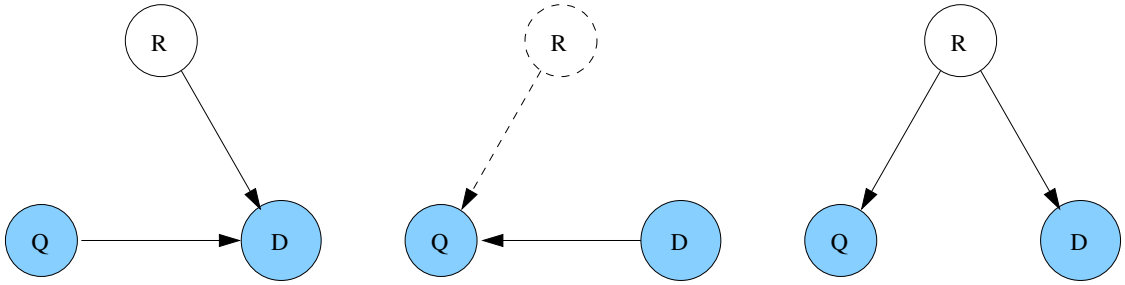


Figure 2.1. Graphical diagrams showing dependencies between the query Q, the document D and relevance R variables in different probabilistic models of IR. Left: classical probabilistic model [113]. Middle: language modeling framework [102] according to [67]. Right: the generative model proposed in this dissertation. Shaded circles represent observable variables.

CHAPTER 3

A GENERATIVE VIEW OF RELEVANCE

In this chapter we will introduce the central idea of this thesis – the idea that relevance can be viewed as a stochastic process underlying both information items and user requests. We will start our discussion at a relatively high level and gradually add specifics, as we develop the model in a top-down fashion.

3.1 An Informal Introduction to the Model.

Suppose we find ourselves in the following environment. We have a user, or a group of users, performing a particular task. In order to perform that task effectively, the user needs to have access to some specific information. This need of information will be represented as an abstract concept $\mathcal{R}$. We also have a collection $\mathcal{C}$ of information items, perhaps in textual form, perhaps as audio files, images, videos, or even as a combination of these forms. We will refer to these items as *documents* D. We assume that a subset of documents in this collection are *topically relevant* to the information need $\mathcal{R}$, which means that if the user examined those documents, he would judge them to discuss the subject matter that is directly related to his need. These documents form the *relevant* subset $\mathcal{C}_{\mathcal{R}}$ of the collection $\mathcal{C}$. It is also natural to allow the user to express his perception of the information need $\mathcal{R}$. We will interchangeably use the words *request* and *query* to refer to these expressions, and will denote them as Q. We recognize the fact that the user may generate multiple requests for the same information need. We will place no restriction on the number or form of these requests, other than that they be consistent. By consistency we mean that the user looking at a given request should be able to determine whether he could have generated that request for his current information need $\mathcal{R}$. Assuming consistency allows us to define the set $\mathcal{Q}_{\mathcal{R}}$ that contains all requests that could have been produced by our user in response to his information need $\mathcal{R}$. With the above definitions, we are prepared to state the major assumption that serves as the foundation for most of the work done in this thesis. We choose to formulate this assumption in the form of a hypothesis:

> **The Generative Relevance Hypothesis (GRH)**: for a given information need, queries expressing that need and documents relevant to that need can be viewed as independent random samples from the same underlying generative model.

In effect what we are stating is that documents in the relevant set $\mathcal{D}_{\mathcal{R}}$ are all alike, they can be described by some stochastic process associated with $\mathcal{R}$. Furthermore, any request $Q \in \mathcal{Q}_{\mathcal{R}}$ a user might formulate for his information need is expected to follow the same stochastic process. We will use the term **relevance model** to refer to the distribution associated with the information need $\mathcal{R}$.

At this point we expect to encounter a lot of skepticism from a savvy IR practitioner. The generative hypothesis seems to imply that documents and queries are identically distributed, so a document relevant to a given query might look a lot like that query. That, of course, is an unreasonable assumption. For instance, if we are dealing with a websearch scenario we expect a typical query to be a set of two, at most three keywords carrying no syntactic structure whatsoever. On the other hand, a relevant document is expected to be a complete webpage, hopefully containing more than the three words that the user typed into the search box. We expect the text in the webpage to be comprehensible, with well-formed grammar and a lot of "glue" words that are never expected to occur as keywords in any query. In addition, a relevant webpage may contain images, embedded audio, incoming and out-going hyper-links and meta-data, such as the title, the author, date of creation, perhaps even XML tags encoding its semantics. It certainly does not seem plausible that a rich representation like this could follow the same underlying distribution as a two-word query. If nothing else, the space that embodies documents appears to be different than the space containing user queries.

3.1.1 Representation of documents and requests

To circumvent the above problem we are going to take a somewhat radical step – we are going to assume that queries and documents originate in some space which is rich enough to represent the attributes of either. So a query, in its latent, unobserved form, contains all the attributes that a document might contain. What these attributes are depends on the specific collection we are dealing with, and on what aspects of the data we are trying to capture with our model. Let us consider a few examples. If we are dealing with a text collection, the latent form of the query includes a full-blown, coherent narration concerning the subject of the underlying information need. If we are working with images, the query embodies a complete bitmap, along with elaborate textual description of what is pictured in that bitmap. Note that in this case, we assume that the documents (images) also contain this textual description, even if in reality we are dealing with a collection of images that are not annotated in any way. Similarly, in a video archive, both documents and queries contain a sequence of bitmap frames, the audio signal, augmented with a complete textual transcript of speech in the audio and of objects and actions in the frames. As another example, consider a question answering scenario: in this case we assume that our representation space consists of pairs that combine a given question with a correct answer. This, again, is a latent or hidden representation of the data.

At some point before they are presented to us, both documents and queries undergo a certain deterministic transformation that converts them to the form that we actually observe. For example, in the case of a text-only collection, the documents remain unchanged, but for the queries the transformation cuts out all the syntactic glue, removes duplicates and discards all but two or three salient keywords. If we are dealing with a collection of images, the transformation would apply to both documents and queries. Documents would be stripped of their textual description, leaving only the bitmap form. For queries, the transform would drop the image portion and reduce the description to a short list of keywords, as we described above. Similarly, in a video collection, documents would keep the frames and the audio signal, queries would keep some words from the transcript or the textual narrative. Finally, in a question-answering scenario questions would be stripped of their answers to form queries, and answers alone would become documents.

3.1.2 Advantages of a common representation

We admit, the above description assumes a fairly convoluted process for a simple fact of a few keywords making their way into the search box. On the other hand, we gain several distinct advantages by hypothesizing a process described above. These are:

1. **We can define a common generative model.** By assuming that documents and queries originate in the same space, we pave the way for defining a single distribution that can describe both documents and queries. This is an absolute necessity if we want to entertain the generative hypothesis.

2. **Anything can be used as a query in the model.** The fact that the query has an artificially enriched latent representation means that any aspect of that representation can initiate the searching process, as long as it happens to be observable. For example, we talked about image queries originating as a bitmap and narrative and then being cut down to a couple of keywords. But if the user happens to provide a sketch of what he is looking for, or perhaps a fragment of a similar image, that information can be used just as well in place of keywords. In fact, a fragment can be used alongside the words if the user happens to provide both. After all, both are just two pieces of the same incomplete puzzle: the document-like latent representation of the query.

3. **The model has natural extensions to multimedia and semi-structured retrieval.** It is almost self-evident that the proposed representation makes it very easy to model such tasks as cross-language or multi-language retrieval, multi-media retrieval or retrieval from semi-structured databases. For example, consider the cross-language retrieval problem, where documents are written in language A and queries issued in language B. In our approach, the latent representation of both documents and queries is a parallel narrative in both languages. During the transformation, documents lose narrative B, and queries are reduced to a few keywords from narrative B. For multi-language retrieval, the latent representation of each document/query includes a complete narrative in all languages involved. For semi-structured retrieval, each representation contains all the fields that are defined in the database schema, and, naturally, any subset of fields can be taken as observable and used as a query. In this respect,

our model is markedly different from previous approaches to these problems. In the past research, most authors had to define completely new retrieval models to handle cross-language or multi-media scenarios. Two notable exceptions to this rule are the *inference network model* proposed by Turtle and Croft [131, 130], and the *translation model* of Berger and Lafferty [11]. Both models provide for graceful extensions to cross-language retrieval and allow some multi-media processing.

4. **The approach is robust for incomplete representations.** An important feature at the core of our approach is a single generative model underlying both the documents and the queries. As a natural side effect, we will be forced to construct a joint distribution over all forms of media present in a given document, or over all fields in a schema, if we are dealing with partially structured documents. This side effect becomes particularly important in semi-structured databases with missing or garbled information. In these cases having a joint generative model is a great advantage, since it will allow us to recover or fill in the missing fields based on the fields that were available. This recovery is very natural in the model: we simply assume that lost or garbled data is part of the transform that takes documents and queries from their latent space to the observable representation. We are not aware of many other models of semi-structured retrieval that provide for a similar recovery mechanism.

5. **Relevance feedback is a natural part of the model.** In a typical retrieval situation the only piece of information available to the system is the user's request, which is a single element of $\mathcal{Q}_{\mathcal{R}}$. However, if it happens that the user is willing to provide another formulation of his request, or perhaps a sample relevant document, the generative hypothesis will allow us to handle the situation gracefully. Recall that the whole idea behind the GRH is that both user's requests ($\mathcal{Q}_{\mathcal{R}}$) and relevant documents ($\mathcal{D}_{\mathcal{R}}$) are random samples from the same underlying model. Getting more examples of either relevant documents or requests will allow us to estimate the model with greater accuracy. However, we want to stress that requests may be used for feedback only if they reflect the current information need $\mathcal{R}$. This is slightly different from the query relevance feedback approach explored by Nallapati, Croft and Allan [93], where relevance of a given document to multiple queries was used to adjust that document's language model. In their case, different queries did not have to represent expressions of the same information need since they were estimating a model for that document only, as opposed to the *relevance* model for a particular information need.

6. **We do not have to factor the probability space.** To clarify this point, recall the classical probabilistic model of Robertson and Sparck Jones (section 2.3.2). The query Q was not a formal part of the model, it was introduced only as a set of assumptions (section 2.3.2.2) to help estimate probabilities without relevant examples. Why was Q not included in the model? It turns out that formally introducing the query as a variable is not as simple as it may seem. Robertson and colleagues made a number of attempts, the best known is perhaps the *unified model* described in [111, 112]; for a more recent attempt see [14]. In every case the authors instantiate two distinct spaces: the space $\mathcal{Q}$ for queries and the space $\mathcal{D}$ for documents. A probabilistic model is then defined over the product space $\mathcal{Q}\times\mathcal{D}$. This seems fine on the surface, but in reality poses a very serious problem: queries and documents live in orthogonal dimensions of the space. The event that word "Washington" appears in the query has nothing in common with the event that the same word is frequent in some document. The former event concerns the $\mathcal{Q}$ part of the space, the latter is part of $\mathcal{D}$. There is no built-in notion of word overlap; it either has to be learned by observing relevant pairs $\{D, Q\}$, or else we have to forcibly introduce *links* between dimensions in $\mathcal{D}$ and dimensions in $\mathcal{Q}$. Introducing these links is also not always a simple matter: Robertson [104, 105] discovered that working in the factored space can lead to certain consistency problems. This prompted him to argue for the necessity of imposing some sort of structure on top of $\mathcal{Q}\times\mathcal{D}$. Without delving further into the details, we would like to stress that the problems described above are the direct result of using the Cartesian product $\mathcal{Q}\times\mathcal{D}$ (factored space). The problems do not exist if we represent queries and documents in the same space. In our model, a word occurring in the query is the same type of event as the word occurring in any document.

3.1.3 Information retrieval under the generative hypothesis

In the previous section we argued for a common document / query space that would pave the way for applying the generative relevance hypothesis. In this section we will describe how the hypothesis can be turned into an operational information retrieval system. Let's assume a simple scenario – we have a collection $\mathcal{C}$ and user's query Q. Our goal is to rank the documents in the collection such that relevant documents end up as high in the ranking as possible. We are going to accept the generative hypothesis and assume that all relevant documents are samples from the same underlying relevance model as the query. Of course we know nothing about this model, and the user provides us neither with examples of relevant documents nor with alternative re-formulations of the same query. How are we to proceed? First, we have to define the appropriate space $\mathcal{S}$ to represent documents and queries. We can do this by examining the documents in the collection and deciding which aspects of the data we want to capture. In the simplest case, the documents will contain plain unstructured text in the same language as the query. In this case we represent everything as sequences of words from the vocabulary of that language. Now we can try to define the relevance model – the generative process that describes relevant documents. This relevance model is a probability distribution over our representation space $\mathcal{S}$; we will denote it by $P_{\mathcal{R}}$. Intuitively, $P_{\mathcal{R}}(x)$ tells us how likely it is that a relevant document would have representation $x \in S$. We have no idea what $P_{\mathcal{R}}$ looks like, and what kinds of representation it favors, but we do know one thing. According to the generative hypothesis, user's query Q is a random sample from the relevance model $P_{\mathcal{R}}(\cdot)$. We can use this fact to get some idea of what $P_{\mathcal{R}}$ might look like. Specifically, we are going to assume that the relevance model is expressed by a certain (quite general) formula with some unknown parameters. To estimate these parameters we will need a sample from $P_{\mathcal{R}}(\cdot)$, and the user's query Q will play the role of that sample. Once we have an estimate of $P_{\mathcal{R}}$, we have some idea of what a relevant document might look like, or at least we know which representations are more likely under the relevance model.

In some sense, the relevance model $P_{\mathcal{R}}(d)$ plays the same role as the probability $P(d|R{=}1)$ in the classical probabilistic model of Robertson and Sparck Jones (section 2.3.2). This analogy instantly gives us one idea for ranking documents: Robertson's probability ranking principle (section 2.3.1). If our relevance model $P_{\mathcal{R}}$ indeed reflects the distribution of relevant documents, then we would achieve optimal ranking by ordering the documents according to the ratio $\frac{P(d|R=1)}{P(d|R=0)}$. We have to find a way to compute the denominator, but that should be fairly straightforward. As we argued before, almost all documents in the collection are not relevant, so we can use collection-wide statistics to estimate non-relevant probabilities $P(d|R{=}0)$. We have tested this form of ranking, and found that it does work reasonably well. However, we will argue later in this chapter that using the probability ratio $\frac{P(d|R=1)}{P(d|R=0)}$ for ranking may not always be optimal. This sub-optimality does not imply a flaw in the probability ranking principle; rather, it is a consequence of certain dependence assumptions we will make about $P_{\mathcal{R}}(\cdot)$. We will demonstrate that under the assumptions we make, a better form of ranking is the relative entropy between the distribution $P_{\mathcal{R}}(\cdot)$ and the distribution induced by the document d under the same sampling process that we used to induce $P_{\mathcal{R}}$ from the user's query Q.

3.2 Formal Specification of the Model.

We will now try to formally define the ideas presented in the previous section. We will start with the most general case and then outline specific representations for the various retrieval scenarios. Before we begin, we would like to stress that our formalism assumes a closed universe: there is a single user, performing a single task with a single underlying information need. Multiple queries may be involved, but they are all expressions of the same underlying need. Relevance is defined with respect to that information need only.

3.2.1 Representation of documents, queries and relevance.

We assume that in their unobserved form both documents and queries can be represented as a sequence of M random variables $X_1 \ldots X_M$. Each X_i takes values in some appropriate space $\mathcal{S}_i$, so the joint representation space is the Cartesian product $\mathcal{S}{=}\mathcal{S}_1 \times \ldots \times \mathcal{S}_M$. In preparation for what follows, it may help the reader to think of $\mathcal{S}$ as the space that contains the most complete representation possible for any given document or

query – a kind of information space, or knowledge space. This, of course, is only an analogy; we make no attempt to represent meaning.

3.2.1.1 Document and query generators

We define a *document generating transform* D to be a function that converts a complete representation $\{x_1 \ldots x_M\} \in S$ to a form in which a document might actually be observed in the collection; formally $D : \mathcal{S} \rightarrow \mathcal{D}$. Similarly, we define the function $Q : \mathcal{S} \rightarrow \mathcal{Q}$ to be the *query generating transform* – a mapping from representations in the complete space $\mathcal{S}$ to something that looks like a user query. We would like to stress that there is no randomness involved in either $D(x)$ or $Q(x)$ – they are normal deterministic functions like $\log(x)$ or $\sqrt{x}$. And for reasons to be detailed later we will want these functions to take a particularly simple form. The main effect of D and Q will be that they take the sequence $x_1 \ldots x_M$ and remove some of the variables from that sequence. The result of either D or Q will be a new shorter sequence $x_{i_1} \ldots x_{i_m}$ that contains some of the elements of the original representation. In a sense, D and Q discard parts of the rich representation that are not supposed to appear in documents and queries respectively. For example, D may strip images of their captions, and Q may remove all but a handful of keywords from a long detailed narration. As a consequence, we observe that the document space $\mathcal{D}$ and the query space $\mathcal{Q}$ are both contained in the full representation space. To get from $\mathcal{S}$ to $\mathcal{Q}$ all we have to do is lose a few dimensions. Note that spaces $\mathcal{D}$ and $\mathcal{Q}$ may or may not overlap.

3.2.1.2 Relevant documents

By a *collection* $\mathcal{C}$ we will mean a finite set of points in the document space $\mathcal{D}$ – the points that correspond to documents that we have in our database. We define the *relevance judgment* to be the function $R : \mathcal{D} \rightarrow \{0, 1\}$ that specifies whether a given document d is going to be judged relevant or non-relevant. We assume that R is deterministic, but cannot be observed unless the user provides us with relevant examples. The relevance function R involves only documents, a query is unnecessary since we are operating in a closed universe with a single underlying information need. The space of all documents that would be judged relevant if they existed is $\mathcal{D}_R = \{d \in \mathcal{D} : R(d)=1\}$. Relevant documents that actually exist in our collection form the *relevant set* $\mathcal{C}_R = \mathcal{D}_R \cap \mathcal{C}$.

3.2.1.3 Relevance in the information space

A reader particularly interested in the nature of relevance may wonder why we chose to define relevance over the space $\mathcal{D}$ of observable documents, when in fact we have a much more general *information* space $\mathcal{S}$. Any assessor would certainly find it easier to judge relevance based on the complete, uncut representations, say an image with a detailed narrative, as opposed to just a bitmap. However, we want our model to reflect reality whenever possible, and in reality a relevance assessor only has access to the observable space $\mathcal{D}$. Furthermore, since the document generating function D is deterministic, our relevance function R automatically induces a similar relevance function $R_{\mathcal{S}} : \mathcal{S} \rightarrow \{0, 1\}$ over the information space; the function is defined as $R_{\mathcal{S}}(x) = R(D(x))$. The relevant subspace of the information space is defined as $\mathcal{S}_R = D^{-1}(\mathcal{D}_R)$. It consists of all representations $x \in \mathcal{S}$ that would be judged relevant once they got transformed into documents $d = D(x)$.

3.2.1.4 Relevant queries

Now that we have defined a notion of relevance on the information space $\mathcal{S}$, we can talk about what it means to be a *relevant request*; i.e. a request expressing the underlying information need. Getting a sample of what a relevant request *might* look like is actually very straightforward in our model. We already know what portion of the information space would be judged relevant ($\mathcal{S}_R = D^{-1}(\mathcal{D}_R)$), so we can simply apply the query generating function $Q(x)$ to every point x in $\mathcal{S}_R$ to get the corresponding set $\mathcal{Q}_R$ of all queries that reflect the information need. So, if we wanted to know if some request q was relevant or not, we could look at the inverse $Q^{-1}(q)$ and check to see if it falls into $\mathcal{S}_R$ or not. However, we might ask whether $Q^{-1}(q)$ can really tell us anything about relevance. The answer is probably no, and the reason hides in the fact that the function Q mapping representations to queries is not a one-to-one mapping. By its very nature Q is a many-to-one

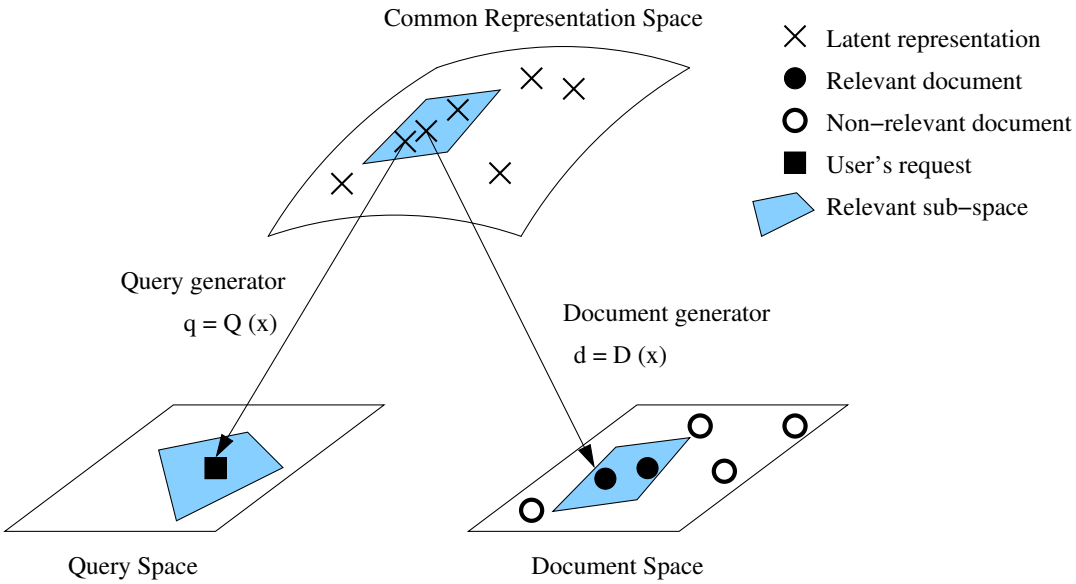


Figure 3.1. Representation of documents, queries and relevance in our model. We assume a latent "information" space which has a relevant sub-space. Documents and queries are deterministic functions of that latent space. Relevant documents and queries are samples from the relevant sub-space.

function: as we discussed above, the effect of Q is to discard some dimensions of the information space, so all representations x that differ on the discarded dimensions will be collapsed into the same query $q = Q(x)$. For example, if the effect of Q was to retain only the first word in a narrative, all possible texts starting with "the" would be collapsed to the same query. This means that the inverse Q^{-1} is not a function, it is a one-to-many mapping. For any given request q there will be many points in the information space, some of them might fall into $\mathcal{S}_R$, most probably will not. It is not possible to definitively say whether any given request is relevant. Fortunately, this inability is not at odds with reality: any human assessor would be hard-pressed to state whether or not a typical query definitively represents the underlying information need. He may say that a request is likely or not likely to express his need; in our model this "likelihood" would be reflected by the amount of overlap between $Q^{-1}(q)$ and the relevant sub-space $\mathcal{S}_R$.

3.2.1.5 Summary of representations

Before we dive into the probabilistic aspects of relevance, we would like to provide a brief summary of the representation choices we made in our model. An informal diagram of these choices is provided in Figure 3.1. We made the following assumptions:

1. Documents and queries originate in the same latent representation space $\mathcal{S}$. We can think of $\mathcal{S}$ as *information* or *knowledge* space; it is rich enough to encode all desired attributes of documents and queries (including non-existent attributes).

2. A query is the result of applying a function Q to some point x in the information space. $Q(x)$ filters and shuffles components of x to make it look like a query. To get a document, we apply a different transform $D(x)$. Both D and Q are deterministic.

3. A user's information need $\mathcal{R}$ corresponds to the *relevant sub-space* of $\mathcal{S}$. Relevant documents and requests result from applying D and Q to points in that sub-space.

3.2.2 Distributions based on relevance

We now have all the building blocks which are required to provide a formal definition for the Generative Relevance Hypothesis (GRH) which was informally introduced in section 3.1. Recall that the hypothesis

states that for a given information need $\mathcal{R}$ there exists a generative model that is a common source of all queries expressing $\mathcal{R}$ and all documents relevant to $\mathcal{R}$. In our case, both documents and queries are deterministic transformations of the information representations $x \in \mathcal{S}$. This fact makes it natural to define the generative model on the space of complete representations, and then extend it to documents and queries using the functions D and Q.

3.2.2.1 Relevance distribution over the information space

Let $P(\cdot)$ denote the joint probability distribution over the variables $X_1 \ldots X_M$. We are particularly interested in the part of $P(\cdot)$ that corresponds to *relevant* points in the information space $\mathcal{S}$. We will refer to this part of $P(\cdot)$ as the *relevance model*, and will define it as: $P_{\mathcal{R}}(\cdot) = P(\cdot|R{=}1)$. If R happened to be fully observable, than we could get the relevance model quite simply by restricting P to the space of all relevant representations $\mathcal{S}_R$. However, R is never fully observable. Even if we have access to complete relevance judgments for a given collection $\mathcal{C}$, we still wouldn't know the value of R for all *hypothetical* documents that make up the document space $\mathcal{D}$. Since we can never know the relevant sub-space $\mathcal{S}_R$, we will define $P_{\mathcal{R}}(\cdot)$ over the entire space $\mathcal{S}$.

In order to make estimation tractable, we will have to make a certain simplifying assumption about the random variables $X_1 \ldots X_M$ that make up our information space. The first thing that comes to mind is to follow the classical probabilistic model and assume that X_i are mutually independent given that $R{=}1$. However, independence is a rather strong assumption, and it will needlessly restrict our model. Instead, we are going to assume that $X_1 \ldots X_M$ are *exchangeable*. To illustrate the concept of exchangeability, let us suppose for a moment that all X_i take values in the same space $\mathcal{S}_1{=}\mathcal{S}_2{=}\ldots{=}\mathcal{S}_M$. Informally this might mean that all X_i are words from the same vocabulary, or that all X_i are numbers in the same range. When we say that X_i are *exchangeable*, we mean that the *order* of observations does not matter. For example, if X_1, X_2, X_3 represent the first, second and third words in a phrase, assuming exchangeability means that the phrase "New York Yankees" will have the same probability as "York Yankees New", and so will any other re-ordering of these words. Exchangeability of the variables does not in any way imply their independence. We can have very strong dependencies between the words[1], as long as these dependencies are not affected by the order in which the words appear.

If we want to be absolutely precise in our definitions, we are dealing with *partial* exchangeability, which means that we can arbitrarily re-order some of the variables, but not all of them. We will be concerned with a very restricted form of partial exchangeability, necessitated by the fact that some variables may take values that are incompatible with other variables. For example, suppose we are dealing with an image database where $X_1 \ldots X_{k-1}$ represent some visual features of the image, and $X_k \ldots X_n$ represent the textual description. In this case we cannot allow any ordering that assigns an image feature ($x_{i<k}$) to a word variable ($X_{j \geq k}$). At the same time, we do allow any re-ordering within $X_1 \ldots X_{k-1}$, and within $X_k \ldots X_n$. For all practical purposes, this restricted form of partial exchangeability is no different from complete exchangeability; all major theoretical results hold without modification.

According to De Finetti's representation theorem, if $X_1 \ldots X_M$ are exchangeable, their joint distribution can be expressed in the following form:

$$P_{\mathcal{R}}(X_1{=}x_1 \ldots X_M{=}x_m) = \int_{\Theta} \left\{ \prod_{i=1}^{M} P_{\theta,\mathcal{S}_i}(x_i) \right\} p(d\theta) \tag{3.1}$$

The variable θ in equation (3.1) is a vector of hidden parameters. If we knew the exact value of θ, the variables X_i would be mutually independent, hence the joint probability decomposes into a product of the marginals $P_{\theta,\mathcal{S}_i}(x_i)$. But θ is unknown, so we integrate over the space of all possible parameter settings Θ. The quantity $p(d\theta)$ is a probability measure that tells us which parameter settings are a-priori more likely to produce a relevant point in $\mathcal{D}$. Each $P_{\theta,\mathcal{S}_i}(x_i)$ is an appropriate probability distribution over one dimension $\mathcal{S}_i$ of our representation space $\mathcal{S}$. Note that if all dimensions S_i are the same (all words, or all numbers), then X_i are also identically distributed.

[1] Consider the famous *contagion* model. We have an urn with two words "a" and "b". We pull out a word, duplicate it and put both duplicates back into the urn. The result of the second draw is certainly not independent of what word we pull out on the first draw: if we pull out "a" than pulling it out again is twice as likely as pulling out "b". But the joint probability of a given sequence of observations will not depend on the order in which they appear: $P(a,a,b) = \frac{1}{2}\frac{2}{3}\frac{1}{4}$ is the same as $P(b,a,a) = \frac{1}{2}\frac{1}{3}\frac{2}{4}$, etc.

3.2.2.2 Relevance distribution over documents and queries

In the previous section we defined a relevance model for representations in the information space $\mathcal{S}$. In this section we will show how the relevance model can be extended to observable documents and queries. As we discussed in section 3.2.1.1, documents and queries are deterministic transformations of information representations $x \in \mathcal{S}$. So we can imagine the following two-step process for generating a relevant document d: first, we pick a relevant representation $x \in \mathcal{S}$ with probability $P_{\mathcal{R}}(x)$; then we apply the document generating transform to get the observation $d = D(x)$. However, since D is not a one-to-one function, there may be many different representations x mapping to the same document d (think of all possible textual descriptions that could go together with a given image). So to get the overall probability of a getting a particular document d, we need to take into account all representations x that could be collapsed into d:

$$P_{\mathcal{R}}(D(X)=d) \;=\; P_{\mathcal{R}}(X \in D^{-1}(d)) \;=\; \sum_{x:D(x)=d} P_{\mathcal{R}}(X=x) \tag{3.2}$$

We will apply the same generative process to explain where queries come from. To get a query q expressing our information need $\mathcal{R}$, we first sample a relevant representation $x \in \mathcal{S}$ with probability $P_{\mathcal{R}}(x)$ and then convert it to a query by setting $q = Q(x)$. And as was the case with documents, the overall probability of getting q depends on all points x that would be mapped to q, so:

$$P_{\mathcal{R}}(Q(X)=q) \;=\; P_{\mathcal{R}}(X \in Q^{-1}(q)) \;=\; \sum_{x:Q(x)=q} P_{\mathcal{R}}(X=x) \tag{3.3}$$

In general, if Q and D were arbitrary functions the formulas for $P_{\mathcal{R}}(d)$ and $P_{\mathcal{R}}(q)$ could end up being quite complex, and in fact might look very different from each other. But recall that in section 3.2.1.1 we restricted D and Q to a particularly simple form. The document transform D takes a representation $x = x_1 x_2 \ldots x_M$ and strips out the components that are not supposed to appear in documents. The result is a sub-sequence $x_{i_1} x_{i_2} \ldots x_{i_n}$, where $n < M$. The query transform Q performs the same type of operation resulting in a sub-sequence $x_{j_1} x_{j_2} \ldots x_{j_m}$, $m < M$. Below we are going to show that restricting D and Q in this fashion leads to a very important equivalence: the relevance model of documents $P_{\mathcal{R}}(d)$ will take the same functional form as the relevance model of queries $P_{\mathcal{R}}(q)$. Furthermore, the functional form will be the same as the relevance model for information representations $P_{\mathcal{R}}(x)$. Below is the derivation concerning the documents. Suppose D removes components $k_1 \ldots k_{M-n}$ from the information representation, and keeps components $i_1 \ldots i_n$. Then the probability of observing a document $d = d_1 \ldots d_n$ is:

$$\begin{aligned}
P_{\mathcal{R}}(D=d) &= \sum_{x \in \mathcal{S}} \delta(d, D(x)) \cdot P_{\mathcal{R}}(X=x) \\
&= \int_\Theta \left\{ \sum_{x \in \mathcal{S}} \delta(d, D(x)) \prod_{i=1}^{M} P_{\theta,\mathcal{S}_i}(x_i) \right\} p(d\theta) \\
&= \int_\Theta \left\{ \sum_{x_1 \in \mathcal{S}_1} \cdots \sum_{x_M \in \mathcal{S}_M} \delta(d_1, x_{i_1}) \cdots \delta(d_n, x_{i_n}) \cdot P_{\theta,\mathcal{S}_1}(x_1) \cdots P_{\theta,\mathcal{S}_M}(x_M) \right\} p(d\theta) \\
&= \int_\Theta \left\{ \underbrace{\prod_{a=1}^{n} \left(\sum_{x_{i_a} \in \mathcal{S}_{i_a}} \delta(d_a, x_{i_a}) P_{\theta,\mathcal{S}_{i_a}}(x_{i_a}) \right)}_{\text{dimensions retained by } D} \underbrace{\prod_{b=1}^{M-n} \left(\sum_{x_{i_b} \in \mathcal{S}_{i_b}} P_{\theta,\mathcal{S}_{i_b}}(x_{i_b}) \right)}_{\text{discarded by } D} \right\} p(d\theta) \\
&= \int_\Theta \left\{ \prod_{a=1}^{n} P_{\theta,\mathcal{S}_{i_a}}(d_a) \right\} p(d\theta)
\end{aligned} \tag{3.4}$$

The first step of the derivation comes from the definition of $P_{\mathcal{R}}(D=d)$ in equation (3.2). The only change we made was to introduce the indicator function $\delta(d, D(x))$ which is one when $d = D(x)$ and zero for any x which does not map to d. This gives the same effect as the subscript $x : D(x)=d$ in equation (3.2). In the

second step, we substitute the definition of the relevance model $P_{\mathcal{R}}$ from equation (3.1). We also apply the Fubini-Tonelli theorem to move the integral $\int_{\Theta}$ outside the summation.[2] The third step in the derivation is the result of expanding the information representation x into its full form $x_1 \ldots x_M$. The summation over all $x \in \mathcal{S}$ becomes a nested summation over all dimensions of $\mathcal{S}$. We also unravel the indicator function $\delta(d, D(x))$ to show that d_a must be equal to $x_{i,a}$ for all document components $a=1 \ldots n$. Recall that $x_{i,a}$ is the a'th element in the sub-sequence that results from applying D to the information representation $x_1 \ldots x_M$. In the fourth step of equation (3.4) we move the components $\delta(d_a, x_{i_a})$ and $P_{\theta,\mathcal{S}_{i_a}}(x_{i_a})$ towards their respective summations $\sum_{x_{i_a} \in \mathcal{S}_{i_a}}$. We can do this because these terms only depend on the values x_{i_a}. Then we note that all summations are in effect disjoint, and we can therefore group them into two products: the first involving the dimensions $\mathcal{S}_i$ that are allowed to occur in the documents and the second containing the dimensions that are discarded by D. To get the final step, we observe that the second product involves the summations of $P_{\theta,\mathcal{S}_{i_b}}(x_{i_b})$ over all possible values of x_{i_b}. Since every $P_{\theta,\mathcal{S}_{i_b}}$ is a probability distribution, all summations will equal to one, and the product of $M-n$ ones equals one. Finally, the indicator function $\delta(d_a, x_{i_a})$ will reduce every summation in the first product to a single term, specifically the term $x_{i_a} = d_a$. This gives us the final form of equation (3.4). If we assume that the query transform Q operates in the same way as D and reduces a representation $x = x_1 \ldots x_M$ to a sub-sequence $x_{j_1} \ldots x_{j_m}$, then we can carry out a similar argument to demonstrate that the query relevance model takes the form:

$$P_{\mathcal{R}}(Q{=}q) = \sum_{x \in \mathcal{S}} \delta(q, Q(x)) \cdot P_{\mathcal{R}}(X{=}x) = \cdots = \int_{\Theta} \left\{ \prod_{b=1}^{m} P_{\theta,\mathcal{S}_{j_b}}(q_b) \right\} p(d\theta) \tag{3.5}$$

An important thing to observe is that equation (3.5) has the same form as equation (3.4), and, for that matter, equation (3.1). The only difference is that the product in each case goes over a different set of component dimensions. So we can conclude that the document relevance model $P_{\mathcal{R}}(d)$ has the same general form as the model of relevant requests $P_{\mathcal{R}}(q)$.

3.2.2.3 Significance of our derivations

Why go through all the trouble to get the result stated above? Why is it important that $P_{\mathcal{R}}(d)$ look like $P_{\mathcal{R}}(q)$? The reason goes to the heart of the assumption made in the generative hypothesis. The hypothesis states that queries and relevant documents are samples from the same underlying model. Without a good justification, this claim will draw an uncomfortable amount of criticism from Information Retrieval practitioners. On the surface, the GRH seems to claim that documents and queries will be identically distributed, which is certainly a false assumption. Documents do not look like queries, they have different characteristics, and claiming that they are samples from the same model appears ludicrous. The sole purpose of our argument is to show that the generative hypothesis is not an absurdity. Documents and queries can and will look different under the GRH. In fact, their observable forms can be completely different: a document can be a bitmap and a query can be a string of text. However, as we demonstrated in the previous sections, we can still formally treat them as samples from the same generative model. We force documents and queries to originate in the same space: the "information" space, which is general enough to represent both. Then we use the functions D and Q to transform the information representation in such a way that queries look like queries and documents look like documents. Very conveniently, these transformations do not change the functional form of the distribution. So, despite their different appearance and different characteristics, queries and relevant documents can be formally treated as samples from the same underlying probability distribution.

3.2.2.4 Summary of relevance distributions

Let us summarize the probabilistic view of relevance we developed in the preceding sections. We show an informal diagram of our model in Figure 3.2.

1. We defined the term *relevance model* to mean the probability distribution $P_{\mathcal{R}}(\cdot)$ associated with relevance. $P_{\mathcal{R}}$ is a distribution over the latent representation space $\mathcal{S}$.

[2] We need the Fubini-Tonelli theorem because the parameter space Θ is uncountable and the information representation space $\mathcal{S}$ may in fact be infinite. We believe the theorem is applicable because the integrand is a product of measurable components: $\delta(d_a, x_{i_a})$, $P_{\theta,\mathcal{S}_{i_a}}(x_{i_a})$ and $p(d\theta)$.

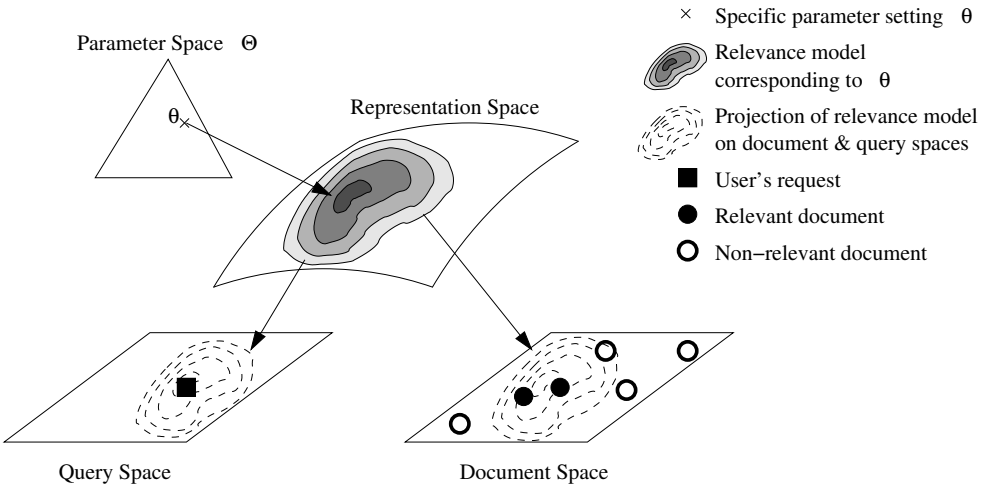


Figure 3.2. A relevance model is a probability distribution over latent representation space $\mathcal{S}$. Dimensions of $\mathcal{S}$ are exchangeable, so probabilities involve a hidden parameter θ. A relevance model has natural projections to observable document and query spaces.

2. We assumed that random variables X_i in the latent representation are *exchangeable* given relevance. By De Finetti's representation theorem, the relevance model $P_{\mathcal{R}}$ will depend on a hidden parameter θ and will take the following form:

$$P_{\mathcal{R}}(x_1 \ldots x_n) = \int_{\Theta} \prod_{i=1}^{n} P_\theta(x_i) p(d\theta)$$

3. We discussed how the document and query transforms D and Q allow us to extend $P_{\mathcal{R}}$ from latent representations to observable documents and queries. We showed that for our choices of D and Q, relevance models of queries and documents will have the same functional form. We used this to justify the generative relevance hypothesis.

3.2.3 Document ranking criteria

Recall that in section 3.1.3 we informally discussed how our model can be used for information retrieval. In this section we will introduce specific formulas that will turn the model into an operational retrieval system. We will discuss three possible forms of ranking that are possible under the generative hypothesis: document likelihood, query likelihood and model comparison. We will also give a brief argument for why we believe the last form of ranking may be preferable.

3.2.3.1 Document likelihood

Let us start by discussing the ranking criterion that was used in the classical probabilistic model of Robertson and Sparck Jones [113]. As we discussed in section 2.3.1, we can achieve optimal retrieval performance if we ranked the documents d according to the posterior probability of relevance $P(R{=}1|D{=}d)$. In section 2.3.2 we explained that the probability of relevance is rank-equivalent to the ratio $\frac{P(D=d|R=1)}{P(D=d|R=0)}$, so theoretically using this ratio should lead to the best retrieval results. In practice the quality of ranking heavily depends on how accurately we can estimate the probability in the numerator. Most previous publications (sections 2.3.2.2 and 2.3.2.3) used heuristic estimates of the probability $P(D{=}d|R{=}1)$ when no relevant examples were available. We will now show that the generative hypothesis makes it possible to compute the estimates in a more formal fashion.

In the previous sections we defined the relevance model $P_{\mathcal{R}}$ to be a distribution associated with relevance. In equation (3.4) we defined the likelihood $P_{\mathcal{R}}(D{=}d)$ of observing a given document d as a result of random

sampling from the relevance model. This likelihood corresponds to the relevant probability $P(D{=}d|R{=}1)$, so we can substitute it into the numerator of the probability ratio and rank the documents by $\frac{P_{\mathcal{R}}(D=d)}{P(D=d|R=0)}$. However, in order to make that substitution we need to specify the last component of equation (3.4): the probability measure $p(d\theta)$. Recall that $p(d\theta)$ tells us which parameter settings θ are more likely to produce a relevant document d. Unfortunately, we have no relevant documents to help us estimate $p(d\theta)$. However, note that the same quantity $p(d\theta)$ appears in equation (3.5), which defines the probability of observing a relevant query. Of course, this is no accident: both equations inherit the same integral from the joint relevance model over latent representations (equation 3.1). The fact that $p(d\theta)$ plays a role in query generation gives us the necessary foothold for estimation.

We are going to proceed in a Bayesian fashion. Let $p_0(d\theta)$ denote some *prior* measure over the parameter space Θ. Naturally, p_0 is the same for all queries and does not reflect any particular information need. We are reasonably certain that the user's query $q = q_1 \ldots q_m$ is relevant, hence we can treat it as a sample from $P_{\mathcal{R}}$ and compute the *posterior* probability measure over the parameter space:

$$p(d\theta|Q{=}q) = \frac{P_{\mathcal{R}}(Q{=}q|\theta)p_0(d\theta)}{P_{\mathcal{R}}(Q{=}q)} = \frac{\left\{\prod_{j=1}^{m} P_\theta(q_j)\right\} p_0(d\theta)}{\int_\Theta \left\{\prod_{j=1}^{m} P_\theta(q_j)\right\} p_0(d\theta)} \tag{3.6}$$

Equation 3.6 gives us the best estimate of $p(d\theta)$ that is possible to achieve given the query q as our only relevant sample. Once we have this estimate, we can proceed in two possible ways. The natural thing to do is to plug $p(d\theta|Q{=}q)$ into equation (3.4) and use that equation to approximate the probability $P_{\mathcal{R}}(d)$ for every document d in the collection. However, this approach will be quite expensive computationally: the integral in equation (3.4) will be non-trivial for all but the simplest probability measures $p_0(d\theta)$. A more tractable approach will be to compute the *expected value* of the parameters θ under our new posterior $p(d\theta|Q{=}q)$:

$$E_q\theta = \int_\Theta \theta p(d\theta|Q{=}q) = \frac{\int_\Theta \theta \left\{\prod_{j=1}^{m} P_\theta(q_j)\right\} p_0(d\theta)}{\int_\Theta \left\{\prod_{j=1}^{m} P_\theta(q_j)\right\} p_0(d\theta)} \tag{3.7}$$

Now we can compute the probability of observing a document $d = d_1 \ldots d_n$ assuming that the hidden parameter θ takes on the value $E_q\theta$:

$$P_{\mathcal{R}}(D{=}d|E_q\theta) = \prod_{i=1}^{n} P_{E_q\theta}(d_i) \tag{3.8}$$

We would like to point out that equation (3.8) does have a formal justification: it happens to be a very close approximation to equation (3.4) when the posterior density $p(d\theta|Q{=}q)$ is highly peaked around some point in Θ. This happens either when the prior $p_0(d\theta)$ is peaked, or when the query q is a long sequence. And as we already mentioned, equations (3.7) and (3.8) are much cheaper computationally, since we have to compute the integral only once per query, as opposed to doing it for every document in the database every time a new query is issued. In all subsequent discussion we will use equation (3.8) as the numerator in the probability ratio $\frac{P(D=d|R=1)}{P(D=d|R=0)}$. The denominator can be computed in a similar fashion using equation (3.8), only this time we would use the *prior* expectation $E_0\theta = \int_\Theta \theta p_0(d\theta)$ in place of $E_q\theta$. Why is this reasonable? Because the prior $p_0(d\theta)$ is going to reflect parameters that are likely on a collection-wide scale, and, as we argued before, the collection as a whole makes for a very good approximation to the non-relevant class. Given the choices we made above, the final ranking criterion would be:

$$\frac{P(D{=}d|R{=}1)}{P(D{=}d|R{=}0)} \approx \frac{P_{\mathcal{R}}(D{=}d|E_q\theta)}{P_{\mathcal{R}}(D{=}d|E_0\theta)} = \prod_{i=1}^{n} \frac{P_{E_q\theta}(d_i)}{P_{E_0\theta}(d_i)} \tag{3.9}$$

3.2.3.2 Query likelihood

If we look at the document-likelihood ranking from a high level, here is what we did: we started with the query q, used it to estimate the parameters of the relevance model $P_{\mathcal{R}}$, and then used $P_{\mathcal{R}}$ to compute the probability of observing a given document d. Diagrammatically, the flow of estimation would look as

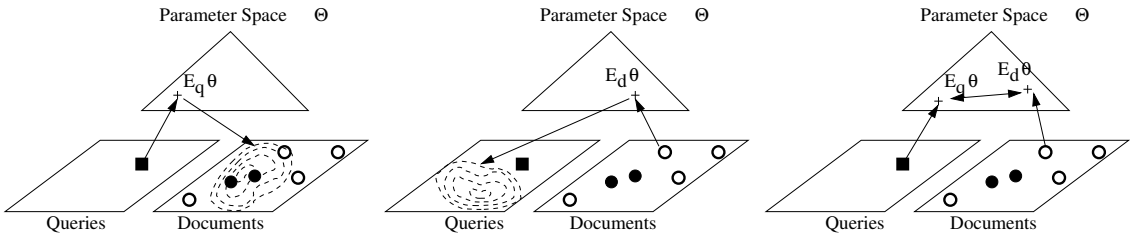
follows: $q \rightarrow P_{\mathcal{R}} \rightarrow d$. Interestingly, we could very easily reverse the flow and compute the probabilities in the opposite direction: $d \rightarrow P_{\mathcal{R}} \rightarrow q$. This is possible because $P_{\mathcal{R}}$ is a symmetric generative model, relevant documents and queries are both random samples from the model, and we could use either to start the estimation process. The only hurdle we face is conceptual: can we estimate the relevance model $P_{\mathcal{R}}$ from a document when we don't know if the document is relevant or not. This was not the case before: we had every reason to believe that user's query was in fact relevant to his information need. A way to get around this problem is to hypothesize that there exists an information need $\mathcal{R}_d$ for which document d is relevant. Then we can use the same Bayesian process as in the previous section to estimate the parameters of $P_{\mathcal{R}_d}$ – the relevance model associated with $\mathcal{R}_d$. We can then rank the documents d by the probability that user's query q would be observed during random sampling from $P_{\mathcal{R}_d}(\cdot)$.

If this formalism looks suspiciously like the language-modeling framework, it should. Conceptually, the document relevance model $P_{\mathcal{R}_d}$ is the same construct as the document language model M_d that was discussed extensively in section 2.3.3. The only difference between the two is that the language-modeling approach assumes the query words to be independent, whereas we assume them to be exchangeable, leading to rather different estimates for $P_{\mathcal{R}_d}$. But if we set technical differences aside, ranking the documents by the probability $P_{\mathcal{R}_d}(Q{=}q)$ would make our model equivalent to the language-modeling approach. If so inclined, we could view language-modeling as a special case of our generative model.

Is query likelihood a good form of ranking? From a large body of publications we certainly know that it is empirically effective. However, it does have two problems. The first is philosophical: in the query-likelihood approach we are not modeling the user's information need in any way. We are hypothesizing a series of needs $\mathcal{R}_d$, one for each document in the dataset, and for each of them we construct a relevance model $P_{\mathcal{R}_d}$. Thus, relevance to user's need is no longer an explicit part of the model, which makes it difficult to define tasks like relevance feedback, and in general abstracts the user from the retrieval setting. See section 2.3.4.1 for a more detailed discussion of the issue. The second issue is computational complexity: estimating a relevance model $P_{\mathcal{R}_d}$ involves computing the posterior $p(d\theta|D{=}d)$ for every document d in the collection. In general, this computation is quite expensive, and repeating it for every new query we get is not an option. Pre-computing and caching this posterior seems like a viable alternative ($p(d\theta|d)$ is independent of q), but we have to keep in mind that Θ is a continuous space, caching a density over this space is not a trivial task. Of course, we could get around the problem by forcing $p_0(\theta)$ be a simple conjugate prior, in this case the integrals would be reduced to closed-form expressions, and everything could be done on-the-fly. The clever trick of taking a simple conjugate prior has been used within the language modeling framework by Zhai and Lafferty [154] when they describe smoothing with Bayesian priors. It was used in a very similar fashion by Zaragoza, Hiemstra and Tipping [152] in the development of what they called the *predictive* language model for IR. In both cases p_0 was the Dirichlet distribution, a conjugate prior to the multinomial distribution which forms the foundation of the language-modeling framework. However, computational convenience has its cost – as we will argue in the following chapter, the simple prior p_0 may not be able to accurately capture interdependencies between components of latent representations. Moreover, we are still left with the philosophical dilemma that query likelihood does not model relevance to the user's information need $\mathcal{R}$.

3.2.3.3 Model comparison

The third ranking criterion we examine is the model comparison approach, which represents a combination of ideas from query likelihood and document likelihood ranking. Recall that for document likelihood, we estimate a relevance model $P_{\mathcal{R}}$ from the query, and then use it to assign a probability to each document. For query likelihood we proceed in the opposite direction, estimating a model $P_{\mathcal{R}_d}$ from each document and using it to compute the probability of observing the query. In the former approach we do a lot of processing on the query side, estimating a non-trivial relevance model $P_{\mathcal{R}}$. In the latter, all estimation is done on the document side. Croft [33] was the first to suggest that a superior form of ranking may involve estimating both the relevance model and the document model, with the purpose of directly comparing the two. Zhai and Lafferty [66, 153] used a similar ranking principle in the development of their risk minimization framework for information retrieval. The principle assumes that both the relevance (query) model $p_q(\cdot)$ and the document model $p_d(\cdot)$ are distributions over the same vocabulary $\mathcal{V}$. In this case, a natural distance metric for comparing the two models would be the Kullback-Leibler divergence $D(p_q||p_d)$, defined as:



Document Likelihood | Query Likelihood | Model Comparison

Figure 3.3. Document ranking criteria. Left: estimate parameters $E_q\theta$ from the query, compute document likelihood. Middle: estimate parameters $E_d\theta$ from the document, compute query likelihood. Right: Estimate parameters from the query, then from the document, compare two estimates using relative entropy.

$$D(p_q||p_d) = \sum_{v \in \mathcal{V}} p_q(v) \log \frac{p_q(v)}{p_d(v)} \tag{3.10}$$

From the information-theoretic standpoint, equation (3.10) describes the *relative entropy* between the two distributions – the number of bits we would waste if we tried to encode distribution $p_q(\cdot)$ using the probability estimates from $p_d(\cdot)$. Note that relative entropy is a *dissimilarity* measure, so we would rank documents d using its negation: $-D(p_q||p_d)$.

In order to use the relative entropy ranking with our generative approach, we would have to express the relevance models $P_{\mathcal{R}}$ and $P_{\mathcal{R}_d}$ in a form that looks like a distribution over the vocabulary. We can do this by constructing an expectation over the parameter vector θ and setting:

$$\begin{aligned} p_q(v) &= \int_\Theta \theta_v p(d\theta|Q{=}q) \\ p_d(v) &= \int_\Theta \theta_v p(d\theta|D{=}d) \end{aligned} \tag{3.11}$$

Where the posterior densities $p(d\theta|q)$ and $p(d\theta|d)$ are computed according to equation (3.6). Once we have these expectations, we can plug them directly into equation (3.10).

3.2.3.4 Discussion of ranking criteria

In the previous sections we discussed three possible criteria for ranking documents under our model: document likelihood, query likelihood and model comparison. Figure 3.3 shows an informal diagram of the three criteria. Each of these has its benefits and drawbacks in terms of effectiveness, computational efficiency and ease of implementation. It turns out that different methods are more natural in different retrieval settings. For example, we will use model comparison in our ad-hoc and cross-language retrieval experiments, query likelihood in multi-media retrieval runs, and a form of document likelihood in topic detection and tracking (TDT). The first choice is motivated entirely by ranking effectiveness, the second is heavily influenced by considerations of computation complexity, and the third arises out of the fact that TDT provides a somewhat unusual environment for which the other choices do not fair well.

The three ranking criteria have strong functional relationships between them, and we would like to briefly discuss the nature of these relationships. Suppose we are dealing with a document d of length n and a query q of length m. We assume that both d and q are mono-lingual strings of text, i.e. they consist of words v from the same vocabulary $\mathcal{V}$. In order to simplify our discussion, we will temporarily adopt the following definitions. Let $\vec{d}$ denote the *empirical* distribution of words in the document, i.e. the distribution we get by counting how many times each word v occurs in d, and the dividing all counts by the document length n. Similarly, let $\vec{q}$ denote the empirical distribution of words in the query. Naturally, all but a handful of dimensions in $\vec{q}$ will be zeroes, $\vec{d}$ may have a few hundred non-zero dimensions. Let $\vec{\theta}_d$ and $\vec{\theta}_q$ be the distributions we get from equations (3.11). Recall that $\vec{\theta}_d$ is the expected value of the parameter vector θ conditioned on observing d as a sample from our generative model. Similarly, we get $\vec{\theta}_q$ by conditioning θ

on the query observation q. Let $\vec{\theta}_0$ denote the expected value of θ under the prior density p_0. The intuitive meaning of $\vec{\theta}_0$ is that of a *background* distribution over the vocabulary – the distribution we would take as a prior before observing anything. Let $\vec{a} \cdot \vec{b}$ denote the inner product of the two vectors $\vec{a}$ and $\vec{b}$. Under these definitions, the three ranking criteria can be re-expressed as follows:

$$\text{Document likelihood: } \frac{P_{\mathcal{R}}(D{=}d)}{P_0(D{=}d)} \propto n\vec{d} \cdot (\log \vec{\theta}_q - \log \vec{\theta}_0) \tag{3.12}$$

$$\text{Query likelihood: } P_{\mathcal{R}_d}(Q{=}q) \propto \vec{q} \cdot \log \vec{\theta}_d \tag{3.13}$$

$$\text{Model comparison: } -D(\vec{\theta}_q||\vec{\theta}_d) \propto \vec{\theta}_q \cdot \log \vec{\theta}_d \tag{3.14}$$

To get the above equivalences we took the following steps. We converted document and query likelihoods to logarithms. We assumed that the posterior densities $p(d\theta|q)$ and $p(d\theta|d)$ are heavily peaked around $\vec{\theta}_q$ and $\vec{\theta}_d$ respectively. Note that the latter assumption is true in most situations; the former is probably false. We omitted the query length m from the query-likelihood equation: it should be $m\vec{q}$, but m is a constant that does not affect document ranking. We also omitted the second term $\vec{\theta}_q \cdot \log \vec{\theta}_q$ from the relative entropy, it does not involve d and thus has no effect on ranking. Expressing the ranking criteria in the form of equations (3.14) allows us to make the following interesting observations:

1. Query likelihood is a special case of model comparison. The only difference between the two is that query likelihood always uses the empirical distribution $\vec{q}$ on the left side, whereas relative entropy ranking allows the left side to be estimated in any way.

2. Document likelihood is heavily influenced by document length n. This can be very problematic for heterogeneous collections, where we can have very short and very long documents. Concatenating a document with itself will double its score, but one would be hard-pressed to argue that concatenation doubles topical relevance.

3. The denominator $P(D{=}d|R{=}0)$ plays an important role in document likelihood. The numerator by itself would favor very short documents consisting mainly of function words (stop-words), as these words are likely to have highest probabilities in any set of documents, including the relevant set.

4. We could perform model comparison in the opposite direction: $-D(\vec{d}||\vec{\theta}_q)$, or more generally: $-D(\vec{\theta}_d||\vec{\theta}_q)$. This corresponds to "encoding" a document with a query model, and would produce results similar to document likelihood, but without the detrimental effect of document length. The closest approximation to document likelihood would be: $D(\vec{d}||\vec{\theta}_0) - D(\vec{d}||\vec{\theta}_q) = \vec{d} \cdot (\log \vec{\theta}_q - \log \vec{\theta}_0)$.

We may observe that model comparison subsumes both document and query likelihood. Query likelihood is always a special case of $-D(\vec{\theta}_q||\vec{\theta}_d)$, while document likelihood is a special case of $D(\vec{\theta}_d||\vec{\theta}_0) - D(\vec{\theta}_d||\vec{\theta}_q)$. Model comparison allows greater flexibility than computing either form of likelihood, since we can carry out parameter inference both for the document and for the query. However, it does raise an interesting question: since we can apply KL divergence in two ways, is one preferable to another? Ultimately, the question will be answered by experimental results; however, we can get some intuition by looking at the functional properties of $D(\cdot||\cdot)$.

3.2.3.5 Query likelihood vs. document likelihood.

Let us attempt an analytical comparison of the two possible ways of applying relative entropy. The first, $(-D(\vec{\theta}_q||\vec{\theta}_d))$, is a generalized form of query likelihood ranking. The second, $D(\vec{\theta}_d||\vec{\theta}_0) - D(\vec{\theta}_d||\vec{\theta}_q)$, is a generalization of document likelihood. Recall that we defined document likelihood to be the probability ratio $\frac{P(D=d|R=1)}{P(D=d|R=0)}$, derived from Robertson's probability ranking principle [110]. Let us assume for the moment that we have the best estimates possible for the vectors $\vec{\theta}_q$ and $\vec{\theta}_d$. That means the probabilities $\theta_q(v)$ accurately reflect the probability of word v appearing in documents relevant to the query q. We are going to argue that in this case, generalized query likelihood is going to lead to better ranking than generalized document likelihood. In order to support this claim, we are going to consider the following question:

Out of all possible documents, which one would be ranked the highest by each method?

First let us consider document likelihood. In mathematical terms, we are looking for a set of parameters $\vec{\theta}_d$, which would maximize the generalized document likelihood. For any given document d, the following inequality holds:

$$\begin{aligned} \text{DL-score}(d) &= D(\vec{\theta}_d||\vec{\theta}_0) - D(\vec{\theta}_d||\vec{\theta}_q) \\ &= \sum_{v \in \mathcal{V}} \theta_d(v) \log \frac{\theta_q(v)}{\theta_0(v)} \\ &\leq 1 \times \log \frac{\theta_q(v^*)}{\theta_0(v^*)} + \sum_{v \neq v^*} 0 \times \log \frac{\theta_q(v)}{\theta_0(v)} \end{aligned} \tag{3.15}$$

where $v^* = \arg\max_v \left\{ \frac{\theta_q(v)}{\theta_0(v)} \right\}$. In other words, to get the highest possible document likelihood score, the document d should place all its probability mass on a single word v^*. The magic word v^* should have a high probability $\theta_q(v)$ of occurring in the relevant documents and a comparatively low probability $\theta_0(v)$ of occurring in the collection at large. So from a document-likelihood standpoint, the perfect document d^* would contain a single word v^*, repeated numerous[3] times. Of course, documents repeating a single word are not likely to occur in any realistic collection. However, the implication of equation (3.15) is this:

> *Document likelihood will favor documents that contain many repetitions of a few highly discriminative words, i.e. words with large values of* $\frac{P(w|R=1)}{P(w|R=0)}$.

Now we will demonstrate that query likelihood, or more generally $D(\vec{\theta}_q||\vec{\theta}_d)$, will favor a very different kind of document. As before, we want to find the parameter vector $\vec{\theta}_d$ that will result in the highest query likelihood score. For any document d we will have:

$$\begin{aligned} \text{QL-score}(d) &= -D(\vec{\theta}_q||\vec{\theta}_d) \\ &= \sum_{v \in \mathcal{V}} \theta_q(v) \log \frac{\theta_d(v)}{\theta_q(v)} \\ &\leq \log \left(\sum_{v \in \mathcal{V}} \theta_q(v) \frac{\theta_d(v)}{\theta_q(v)} \right) \\ &= 0 = D(\vec{\theta}_q||\vec{\theta}_q) \end{aligned} \tag{3.16}$$

The third step in equation (3.16) is Jensen's inequality applied to the expectation of a logarithm; the last step comes from the fact that relative entropy of a distribution with itself is zero. The meaning of equation (3.16) is that the highest-ranking document d^* will distribute its probability mass among words in the same way as the relevance model $\vec{\theta}_q$. As before, such a document d^* may not exist in our collection, but the implication of our argument is this:

> *Query likelihood will favor documents that contain many distinct words which are prevalent in the relevant class, i.e. words for which* $P(w|R=1)$ *is non-zero.*

The conclusion of our analysis is that document likelihood and query likelihood represent two very different ranking principles. They will favor two very different types of documents: the former will prefer a few highly relevant words, the latter will opt for as many possibly relevant words as possible. Our intuition suggests that query likelihood ought to be the better choice: it will tend to select documents d that look like the expected relevant document. However, empirical performance being the ultimate judge, we will test both alternatives in chapter 5.

3.3 Discussion of the Model

This chapter was dedicated to the theoretical development of the generative view of relevance. We provided formal definitions of three components of the model: **(i)** the representation of documents, queries and relevance, **(ii)** the probabilistic distributions associated with relevance, and **(iii)** the criteria for ranking documents

[3] Multiple repetitions of v^* are necessary to make the posterior density $p(d\theta|d^*)$ spiked around the desired point θ^* such that $\theta^*(v^*) \approx 1$ and $\theta^*(v) \approx 0$ for all $v \neq v^*$.

in response to a user's query. In our discussion we skipped one seemingly small component: we never specified the probability distribution $p(d\theta)$ over the parameter space Θ. This omission is by no means accidental. In fact, $p(d\theta)$ plays an absolutely critical part in our model, and we felt it necessary to devote the entire next chapter to its development. In the present chapter we were also guilty of keeping our definitions unreasonably vague. We never really specified the nature of the information space $\mathcal{S}$, and neither did we talk about the meaning of random variables $X_1 \ldots X_M$ that form the latent representation of documents and queries. This too was done on purpose – we wanted to keep our definitions as abstract as possible, so the model would be applicable to a wide range of specific retrieval scenarios. We will provide the complete representation details in chapter 5. We will now take some time to discuss the advantages of our model and highlight differences from previously proposed retrieval models.

The centerpiece of our model is the generative relevance hypothesis. To the best of our knowledge, none of the existing retrieval models makes a similar assumption about queries and relevant documents. The classical model of Robertson and Sparck Jones [113] does assume a common model for relevant documents, but it never considers that user queries could be treated as samples from the same model. On the other hand, the language-modeling approach of Ponte and Croft [102] tests whether the query could be a sample from a *given* document model, but it never conceives that *all* relevant documents might be samples from a common model. Other probabilistic formulations either follow the classical model [113] or the language-modeling paradigm [102]. One noteworthy exception is the risk-minimization framework of Zhai and Lafferty [66], where the authors hypothesize the existence of *two* generative models: one is responsible for generating documents, the other generates queries. There is no overlap between the two models: they do not have to share the same event space, and the process of generating documents can be completely different than the process of sampling queries. Instead of relevance the authors define a more general notion of *risk* on the posterior estimates of the two models.

We believe our model represents the first formal attempt to tie together all the three components of a retrieval system: documents, queries and relevance. Recall that the classical model and its extensions generally focus on the link between relevance and documents. The query exists only as a set of heuristics to help us estimate relevant probabilities, it is never treated as an observation. To contrast that, language modeling approaches focus exclusively on the link between documents and queries. Relevance is not a formal part of the model, and as we discussed in section 2.3.4.1, introducing it may not be a trivial matter. Tying both documents and queries to relevance allows us to avoid heuristics even when no relevant examples are provided. When we do have relevant examples, we treat them just like the query – as samples from the underlying relevance model.

Almost every probabilistic model of information retrieval assumes word *independence* in one form or another. The difference in our model is in that we assume *exchangeability* rather than independence. Does this matter? In light of our discussion in sections 2.3.2.5 and 2.3.3.4 one may be tempted to say that there is no benefit in modeling dependence. We beg to differ. Exchangeability, coupled with our estimates for the generative density $p(d\theta)$ will allow us to do things that cannot be easily done in other models. It will allow us to estimate relevant probabilities using the query alone. It will make massive query expansion a formal part of the model. It will allow us to associate English queries with Chinese documents, let us annotate a video segment with appropriate keywords, or retrieve a bitmap image based on a text query. In our opinion, assuming exchangeability instead of independence may prove to be quite beneficial.

CHAPTER 4

GENERATIVE DENSITY ALLOCATION

The present chapter plays a special role in this thesis. In this chapter we will not be talking about relevance, documents or queries. In fact, this chapter will have very little to do with information retrieval. The subject of our discussion will be generative models for collections of discrete data. Our goal is to come up with an effective generative framework for capturing interdependencies in sequences of exchangeable random variables. One might wonder why a chapter like this would appear in a thesis discussing relevance. The reason is simple: a generative model lies at the very heart of the main assumption in our model. Our main hypothesis is that there exists a generative model that is responsible for producing both documents and queries. When we construct a search engine based on the GRH, its performance will be affected by two factors. The first factor is whether the hypothesis itself is true. The second factor is how accurately we can estimate this unknown generative process from very limited amounts of training data (e.g. a query, or a single document). Assuming the GRH is true, the quality of our generative process will be the single most important influence on retrieval performance. When we assume the generative hypothesis, we are in effect reducing the problem of information retrieval to a problem of generative modeling. If we want good retrieval performance, we will have to develop effective generative models.

4.1 Problem Statement

The previous chapter laid the groundwork for the kinds of generative models we will be dealing with. We are interested in probability distributions for sequences of exchangeable random variables $X_1 \ldots X_n$, each taking values in some event space $\mathcal{S}_i$. In order to make our discussion clearer we will make a following simplification. We assume that random variables X_i represent *words* from a common finite vocabulary $\mathcal{V}$. The sequences $X_1 \ldots X_n$ represent strings or sentences over that vocabulary. Our goal is then to construct an effective model for finite strings of text, keeping in mind that order of the words in these strings does not matter. We would like to stress that we adopt the terms "words" and "text" only as a matter of convenience. The generative models we describe can be applied to any dataset of discrete exchangeable data. Furthermore, in chapter 5 we will show that our models can be generalized to real-valued variables. But for now, let us stick to strings of text, or, more accurately, *bags of words*, since the order of words in a string makes no difference.

4.1.1 Objective

As we mentioned above, our goal is to construct *effective* models of strings. By definition, a generative model is a probabilistic representation of the data. However, there may exist different points of view about what constitutes an effective probabilistic representation; we would like to make explicit what we are aiming for. We are not interested in *compressing* the data in either lossy or lossless fashion, so we do not care whether our representation is compact, or how many parameters it uses. Nor is it in our goals to come up with a grand theory for bags of words, so we are not going look for the most elegant representation. Our goal is to construct a *predictive* model that is able to generalize – i.e. a model that can be trained from a set of sample strings and then accurately predict new, previously unobserved strings. Specifically, we will assume that we have a training collection of strings $\mathcal{C}_{train}$, and a disjoint collection of testing strings $\mathcal{C}_{test}$. Our goal is to use strings from $\mathcal{C}_{train}$ to estimate all necessary parameters of the model, and then have the model predict every string from $\mathcal{C}_{test}$. A model that assigns comparatively higher likelihood to the testing strings will be considered superior, regardless of its form or the number of parameters it uses. We would like to stress that higher likelihood on $\mathcal{C}_{test}$ measures the ability of the model to *generalize* to new data, and not its ability to *overfit*.

4.2 Existing Generative Models

We will start our development by looking at five existing probabilistic models that are commonly applied to discrete exchangeable collections. We will consider consider the following models: **(i)** the unigram model, **(ii)** the mixture model, **(iii)** the Dirichlet model, **(iv)** probabilistic Latent Semantic Indexing (pLSI) and **(v)** Latent Dirichlet Allocation (LDA). The first two models are extremely simple and are frequently employed in a large number of fields. The Dirichlet model can be thought of as a Bayesian version of the unigram model. The last two models represent a somewhat less-known family of *simplicial mixture* models. Both are considerably more complex than the first three models, both require advanced methods of inference for parameter estimation. We will present the five models in order of increasing complexity. In each case we will identify a weak spot of the model – its inability to deal with some peculiarity of the data. Each subsequent model will address the weakness of its predecessor.

A reader familiar with language modeling will notice that our discussion does not include the following three families of models: (i) Markov chains, (ii) probabilistic grammars and (iii) maximum-entropy models. All three are considered to be staples of natural language processing. We do not discuss them because all three are based on the order of words in a string and cannot be applied to exchangeable sequences.

4.2.1 The Unigram model

The unigram model is perhaps the simplest model we can construct. The model consists of a single distribution U over our vocabulary $\mathcal{V}$. It is a vector of probabilities $U(v)$ one for each word v in the vocabulary. Informally we can think of U as an urn that contains $k{\cdot}U(v)$ counts of each word v, where k is some very large constant. To generate strings of text, we randomly pull out a word from this urn, observe its value, and put it back into the urn. The draws are independent of each other, so the probability of getting a particular sequence of words $w_1 \ldots w_n$ is:

$$P_{uni}(w_1 \ldots w_n) = \prod_{i=1}^{n} U(w_i) \tag{4.1}$$

Estimation of parameters for the unigram model is a very simple task. Given a collection of training strings $x \in \mathcal{C}_{train}$, we simply count how many times we observed each word v. The counts are then normalized and some form of smoothing is applied:

$$U(v) = \lambda \frac{\sum_x n(v,x)}{\sum_x |x|} + (1-\lambda) b_v \tag{4.2}$$

Here $n(v,x)$ denotes the number of times v occurred in the training string x, and $|x|$ stands for the length of x. Summations go over training string $x \in \mathcal{C}_{train}$. Smoothing is achieved by interpolating normalized frequencies with some *background* distribution b_v over the vocabulary. In the absence of prior information, b_v is usually assumed to be uniform over $\mathcal{V}$. The constant λ controls the degree of variance in the estimator. Setting $\lambda = 1$ turns equation (4.2) into the *maximum likelihood* estimator, i.e. the distribution U_{ml} which assigns the maximum possible probability to the training collection $\mathcal{C}_{train}$ under equation (4.1). U_{ml} also represents an *unbiased* estimator. By the law of large number, if we repeatedly estimate U_{ml} with random training collections $\mathcal{C}_{train}$, the quantity $U_{ml}(w)$ will eventually converge to the true probability for each word v. However, U_{ml} is a *high-variance* estimator, meaning the estimates $U_{ml}(v)$ will vary a lot if we consider different training collections, especially if these collections are relatively small. Setting $\lambda < 1$ will reduce the variance, but will also introduce a bias in the estimates, i.e. $U(v)$ will no longer converge to true probabilities.

The unigram model is simple, easy to estimate and is the first choice adopted by researchers dealing with discrete collections. However, it does have a serious drawback: it can only model *homogeneous* collections, where all strings are expected to be similar to each other. If we have two strings a and b, the unigram model would not be affected by taking some words from a and swapping them with words from b: the joint probability of a and b would be the same. This becomes a serious problem if string a discussed politics and string b talked about carburetors. We would like our generative model to recognize that political lingo is unlikely to be used in the same sentence as automotive jargon. This is particularly important because our ultimate interest lies in *topical* relevance of strings.

4.2.2 The Mixture model

The mixture model, also known as *cluster* model or *topic* model, represents a natural evolution of the unigram model. It is designed specifically for dealing with *heterogeneous* collections, where strings can talk about multiple topics. In the mixture model we assume that all possible strings fall into a finite set of k clusters. Strings in each cluster discuss a particular topic z, distinct from all other topics. Each topic z is associated with a distribution T_z over our vocabulary. Words v with high probabilities $T_z(v)$ reflect the jargon associated with topic z. The model also includes a mixture distribution $\pi(z)$, which tells us how prominent topic z is in the collection. When we want to generate a new string $w_1 \ldots w_n$ from the mixture model, we carry out the following process:

1. Pick a topic $z \in \{1 \ldots k\}$ with probability $\pi(z)$.
2. For $i = 1 \ldots n$: pick word w_i from topic z with probability $T_z(w_i)$

The overall likelihood of observing $w_1 \ldots w_n$ from the mixture model defined by parameters $\pi(\cdot)$ and $T_z(\cdot)$ is:

$$P_{mix}(w_1 \ldots w_n) = \sum_{z=1}^{k} \pi(z) \prod_{i=1}^{n} T_z(w_i) \tag{4.3}$$

Estimation of parameters in the mixture model is considerably more varied than in the unigram model. We are aware of two general approaches to estimation. The first is based on *clustering* the training strings, the second, more formal approach, is based on maximizing likelihood of $\mathcal{C}_{train}$. In the first approach (e.g. [148]) we cluster strings in the training collection $\mathcal{C}_{train}$ into k groups: $C_1 \ldots C_k$. The groups typically do not overlap, and result from applying any one of the wide range of clustering algorithms [79]. Once we group the strings, we can estimate a topic model T_z for each cluster C_z using equation (4.2), just as we did for the unigram model. However, it is also a common practice to interpolate relative word frequencies from cluster C_z with the overall unigram model U in the following fashion:

$$T_z(v) = \lambda_1 \frac{\sum_{x \in C_z} n(v,x)}{\sum_{x \in C_z} |x|} + \lambda_2 \frac{\sum_x n(v,x)}{\sum_x |x|} + \lambda_3 b_v \tag{4.4}$$

The first summation concerns the strings x that fall into the cluster of the current topic, while the second involves all training strings. The constants λ_1, λ_2 and λ_3, control the overall estimator variance; they should be positive and must add up to 1. The mixing distribution $\pi(z)$ is commonly taken to reflect the relative size of each cluster: $\frac{|C_z|}{|\mathcal{C}_{train}|}$.

The second approach to parameter estimation is somewhat more formal. We assume that the training strings $\mathcal{C}_{train}$ represent a random sample from the generative model defined by equation (4.3), and try to find the distributions $\pi(\cdot)$ and $T_z(\cdot)$ which maximize the log-likelihood:

$$\sum_{x \in \mathcal{C}_{train}} \log \left\{ \sum_{z=1}^{k} \pi(z) \prod_{i=1}^{n} T_z(w_i) \right\} \tag{4.5}$$

The parameters maximizing equation (4.5) cannot be expressed in closed form, but can be approximated using either iterative gradient ascent, or the Expectation Maximization (EM) algorithm. For details of the latter approach see [98].

The mixture model represents a definite improvement over the unigram model, but it has two important limitations. The first has to do with the number of topics. The number k has to be selected *a priori*, before we start either the clustering or the maximization process. If we happen to pick a k that is too small, we will lose some resolution in the model: several distinct topics will be collapsed to the same distribution T_z, leading to the same problem that we encountered with the unigram model. Making k too large will lead to a different problem: our mixture model will have too many parameters, and the estimates will overfit to the training data.

The second limitation of the mixture model is somewhat more subtle. The model assumes that each string is generated by a single topic model T_z. Consequently, the mixture will assign unreasonably low probabilities to strings that discuss two or more distinct topics, e.g. sports and weather, or politics and medicine. This may

not matter for short strings of text, such as single sentences, but leads to serious problems when we start dealing with complete news stories or editorials. In order to model these kinds of documents accurately, we would have to make k so large that it covers any combination of topics that might be prominent in the collection. And, as we mentioned above, making k too large will lead to serious over-fitting to the training data.

4.2.3 The Dirichlet model

The Dirichlet model is our real first example of Bayesian ideas in a generative model. We are going to devote slightly more space to it, because it will play an important role in the development of two subsequent models: Latent Dirichlet Allocation (LDA), and our own model. One way to view the Dirichlet model is as expression of *uncertainty* over the parameters U of the unigram model. Recall that U is a distribution over the vocabulary. At the same time, U is only a single point in the simplex $\mathbb{P}_\mathcal{V} = \{u \in [0,1]^\mathcal{V} : |u|=1\}$. The simplex is simply the set of all possible distributions over our vocabulary. When we adopted the unigram model we placed all our confidence on one point, U. From a Bayesian perspective that reflects extreme confidence in our estimate U. If we want to allow a bit of healthy uncertainty, we should spread our confidence over a number of points. That can be accomplished by introducing a density $p(u)$ over the points $u \in \mathbb{P}_\mathcal{V}$. Using $p(u)$ as a measure of uncertainty would give us the following process for generating a string of text $w_1 \ldots w_n$:

1. Pick a unigram estimate u with probability $p(u)$ from the distribution space $\mathbb{P}_\mathcal{V}$
2. For $i = 1 \ldots n$: pick word w_i from the unigram model u with probability $u(w_i)$

We should immediately recognize that the generative process for the Dirichlet model looks exactly like the process of the mixture model. The only difference is that before we were dealing with a finite number of topics $T_1 \ldots T_n$, whereas now we have an uncountably infinite space of topics $\mathbb{P}_\mathcal{V}$. In a way, the Dirichlet model can be viewed as a curious solution for deciding how many topics we should have: instead of deciding on k, just pick uncountably many. As a result, the summation of equation (4.3) will turn into an integral:

$$P_{dir}(w_1 \ldots w_n) = \int_{\mathbb{P}_\mathcal{V}} \left(\prod_{i=1}^{n} u(w_i) \right) p(u) du \tag{4.6}$$

A natural question might arise: if we had problems with parameter estimation for a finite number k of topics, how can we handle infinitely many of them? The reason is simple: in the mixture model we had to estimate an entire distribution T_z for every topic z. Here every topic u is already defined, we only have to pick the density $p(u)$. And in practice, considerations of computational complexity drive most researchers to pick a simple conjugate prior as the density $p(u)$. The appropriate conjugate prior for the unigram model is the *Dirichlet* distribution:

$$p(u) = \frac{\Gamma(\sum_{v \in \mathcal{V}} \alpha_v)}{\prod_{v \in \mathcal{V}} \Gamma(\alpha_v)} \prod_{v \in \mathcal{V}} u(v)^{\alpha_v - 1} \tag{4.7}$$

Here $\Gamma(x)$ is the Gamma function. The distribution has a set of $|\mathcal{V}|$ parameters α_v, one for each word v in the vocabulary. The parameters α_v can be estimated in closed form by treating the training strings $\mathcal{C}_{train}$ as an observation from equation (4.6) under the prior Dirichlet density where all parameters are set to a constant β. Omitting the derivation, the posterior estimates take a very simple form:

$$\alpha_v = \beta + \sum_{x \in \mathcal{C}_{train}} n(v, x) \tag{4.8}$$

Here $n(v, x)$ is the number of times the word v occurred in the training string x. Curiously, the total number of parameters in a Dirichlet model is the same as in the simple unigram model; one more to be exact, since α_v don't have to add up to one. The small number of parameters hints at the fact that the Dirichlet distribution may not be as complex as it seems. Indeed, we will show that the Dirichlet model *does not* contain an infinite number of topics. In fact it involves only one topic, slightly more sophisticated than the unigram model. If we plug the density $p(u)$ from equation (4.7) into equation (4.6), we will get:

$$P_{dir}(w_1 \ldots w_n) = \int_{\mathbb{P}_\mathcal{V}} \left(\prod_{i=1}^{n} u(w_i) \right) \left(\frac{\Gamma(\Sigma_v \alpha_v)}{\prod_v \Gamma(\alpha_v)} \prod_v u(v)^{\alpha_v - 1} \right) du$$

$$
\begin{aligned}
&= \frac{\Gamma(\Sigma_v \alpha_v)}{\prod_v \Gamma(\alpha_v)} \cdot \int_{I\!P_{\mathcal{V}}} \left(\prod_v u(v)^{\alpha_v - 1 + n(v, w_{1\ldots n})} \right) du \\
&= \frac{\Gamma(\Sigma_v \alpha_v)}{\prod_v \Gamma(\alpha_v)} \cdot \frac{\prod_v \Gamma(\alpha_v + n(v, w_{1\ldots n}))}{\Gamma(\Sigma_v \alpha_v + n)} \\
&= \frac{\Gamma(\Sigma_v \alpha_v)}{\Gamma(\Sigma_v \alpha_v) \cdot (\Sigma_v \alpha_v) \cdot (\Sigma_v \alpha_v + 1) \cdots (\Sigma_v \alpha_v + n - 1)} \\
&\times \prod_v \frac{\Gamma(\alpha_v) \cdot (\alpha_v) \cdot (\alpha_v + 1) \cdots (\alpha_v + n(v, w_{1\ldots n}) - 1)}{\Gamma(\alpha_v)} \\
&= \prod_{i=1}^{n} \frac{\alpha_{w_i} + n(w_i, w_{1\ldots(i-1)})}{\Sigma_v \alpha_v + (i-1)} \qquad (4.9)
\end{aligned}
$$

If we squint really hard at equation (4.9), we will immediately recognize it as something very familiar. In fact, were it not for $n(w_i, w_{1\ldots(i-1)})$ on the top and $(i-1)$ on the bottom, equation (4.9) would be exactly the same as the unigram model (equation 4.1). The proportion $\frac{\alpha_{w_i}}{\Sigma_v \alpha_v}$ would play the role of $U(w_i)$, the probability of observing the word w_i from the unigram urn. What makes the Dirichlet model different is the component $n(v, w_{1\ldots(i-1)})$, which represents the number of times the current word w_i appears among previous words $w_1 \ldots w_{i-1}$. The Dirichlet model captures a very interesting kind of dependency: the dependency of a word on its own previous occurrences. Every appearance of a word v in the string makes subsequent appearances more likely. In this way the Dirichlet model reflects a very important property of natural language: the fact that word appearances are rare but *contagious* events – once we see a word, we are likely to see it again and again. Dirichlet is clearly superior to the unigram model where multiple occurrences of any word are completely independent of each other. However, all interactions in the model are limited to repetitions of the same word; unlike the mixture model, the Dirichlet cannot capture the fact that "carburetor" goes with "cars" but not with "politics". So we are not really justified in viewing the Dirichlet model as a mixture of an infinite number of topics; in fact the Dirichlet model is not capable of capturing more than a single topic.

4.2.4 Probabilistic Latent Semantic Indexing (PLSI)

The probabilistic LSI model was introduced by Hoffman in [58] and quickly gained acceptance in a number of text-modeling applications. In some circles PLSI is known as the *aspect model*, or as a *simplicial mixture* model. Conceptually, the roots of PLSI go back to methods of latent semantic analysis (LSA / LSI, e.g.[40]), which involve singular-value decomposition of a matrix of real-valued observations. One of the main problems with applying LSA to language is that word observations are not real-valued. Natural text is a fundamentally discrete phenomenon, treating it as continuous observations is certainly possible, but constitutes a stretch of assumptions. PLSI was designed by Hoffman as a discrete counterpart of LSI and PCA/ICA techniques. Another way to look at PLSI is as an attempt to relax the assumptions made in the mixture model. Recall that a simple mixture model assumes that each string of text is generated by a *single* topic model T_z. This can be problematic for long strings of text, e.g. articles or editorials, which can discuss more than one subject. Hoffman's solution to the problem was simple and elegant: he allowed each string to contain a mixture of topics. Hypothetically, every word in a string could come from a different topic. Just like the mixture model, the PLSI model is defined by a set of k topic models $T_1 \ldots T_k$, and a mixture distribution $\pi(\cdot)$. The process of generating a given string $\mathbf{w} = w_1 \ldots w_n$ is as follows:

1. Pick a mixing distribution $\pi_{\mathbf{w}}$ corresponding to $\mathbf{w}$.
2. For each position in the string $i = 1 \ldots n$:
   (a) Pick a topic $z \in \{1 \ldots k\}$ with probability $\pi_{\mathbf{w}}(z)$.
   (b) Pick a word w_i from the topic z with probability $T_z(w_i)$

The overall likelihood of observing $\mathbf{w} = w_1 \ldots w_n$ under the above generative process is:

$$P_{lsi}(w_1 \ldots w_n) = \prod_{i=1}^{n} \left(\sum_{z=1}^{k} \pi_{\mathbf{w}}(z) T_z(w_i) \right) \tag{4.10}$$

When we compare the above equation to the mixture model (equation 4.3), two things strike the eye. First, the summation $\sum_z$ moved inside the product – this reflects the fact that topics are mixed inside the string on the level of a single word w_i. The second observation is worrisome: the mixture distribution $\pi_{\mathbf{w}}(\cdot)$ is conditioned on $\mathbf{w}$, the very string we are trying to generate. Technically, such parametrizations are not allowed in a generative model: it makes no sense to condition the probability of an event on itself. In a way, $\pi_{\mathbf{w}}$ represents an *oracle*, which tells us exactly in what proportion we should mix the topics for a particular string $\mathbf{w}$. Such distributions are not allowed because they usually do not add up to one over the space of all possible samples. A common way to turn equation (4.10) into a proper generative model, is to marginalize the model by summing over all strings $\mathbf{w}$ in the training set. A marginalized PLSI model assigns the following probability to a new, previously unobserved string $w_1 \ldots w_n$:

$$P_{lsi}(w_1 \ldots w_n) = \frac{1}{N} \sum_{\mathbf{w}} \prod_{i=1}^{n} \left(\sum_{z=1}^{k} \pi_{\mathbf{w}}(z) T_z(w_i) \right) \tag{4.11}$$

where the outside summation goes over every training string $\mathbf{w}$, and N represents the total number of training strings. When we discuss PLSI in the following sections, we will be referring to the generative version (equation 4.11), not the one based on equation (4.10).

The parameters of PLSI include the topic distributions $T_1 \ldots T_k$, and the mixture distributions $\pi_{\mathbf{w}}$ for every string $\mathbf{w}$ in the training collection $\mathcal{C}_{train}$. Parameters are estimated by assuming that the training strings represent an observation from equation (4.10). The goal is to find the topics $T_1 \ldots T_k$ and mixtures $\pi_{\mathbf{w}}$ that maximize the log-likelihood of $\mathcal{C}_{train}$:

$$\sum_{\mathbf{w} \in \mathcal{C}_{train}} \sum_{i=1}^{n_{\mathbf{w}}} \log \left(\sum_{z=1}^{k} \pi_{\mathbf{w}}(z) T_z(w_i) \right) \tag{4.12}$$

Maximizing of equation (4.12) has to be done iteratively. It is amenable to gradient methods, but a more common approach is to apply expectation maximization (EM); details can be found in [58]. Due to a very large number of mixing distributions $\pi_{\mathbf{w}}$, PLSI is extremely prone to over-fitting. Hoffmann [58] suggests using simulated annealing to temper the maximization process; other authors found it necessary to cluster the training strings, initialize the topic models T_z from clusters C_z, and then run only a handful of EM iterations to prevent over-fitting.

4.2.5 Latent Dirichlet Allocation

The main problem of PLSI comes from inappropriate generative semantics. Tying the mixing distribution $\pi_{\mathbf{w}}$ to individual strings $\mathbf{w}$ makes the model both confusing and susceptible to over-fitting. Recently Blei, Ng and Jordan [12] introduced a new, semantically consistent aspect model, which attracted a tremendous amount of interest from the statistical learning community. They called the model *Latent Dirichlet Allocation*. The basic setup of the model closely resembles PLSI: the authors assume that the data can be explained by a set of k topics, each topic z is associated with a distribution T_z over our vocabulary, and each word in a given string can come from a mixture of these topics. The main contribution of LDA was in the way of defining the mixing distributions $\pi(z)$. Blei and colleagues make π a random variable, which means it can be any distribution over the topics $z = 1 \ldots k$. Formally, any given mixture $\pi(\cdot)$ is a single point in the simplex $\mathbb{P}_k = \{x \in [0,1]^k : |x| = 1\}$, which includes all possible distributions over the numbers $1 \ldots k$. To reflect the uncertainty about the specific value of $\pi(\cdot)$, the authors impose a density function $p_k(\pi)$ over the points $\pi \in \mathbb{P}_k$ in the simplex. Under these conditions, the process of generating a string $w_1 \ldots w_n$ takes the following form:

1. Pick a mixing distribution $\pi(\cdot)$ from $\mathbb{P}_k$ with probability $p_k(\pi)$.

2. For each position in the string $i = 1 \ldots n$:

   (a) Pick a topic $z \in \{1 \ldots k\}$ with probability $\pi(z)$.

   (b) Pick a word w_i from the topic z with probability $T_z(w_i)$

The generative process behind LDA closely resembles PLSI, but has one important difference: instead of illegally taking $\pi_{\mathbf{w}}(\cdot)$ as the topic weighting distribution, the authors randomly sample $\pi(\cdot)$ from the density function $p_k(\pi)$. The likelihood assigned by LDA to a given string $w_1 \ldots w_n$ is:

$$P_{lda}(w_1 \ldots w_n) = \int_{\mathbb{P}_k} \left\{ \prod_{i=1}^{n} \sum_{z=1}^{k} \pi(z) T_z(w_i) \right\} p_k(\pi) d\pi \tag{4.13}$$

As their density function p_k, the authors pick the Dirichlet distribution (equation 4.7) with parameters $\alpha_1 \ldots \alpha_k$:

$$p_k(\pi) = \Gamma\left(\sum_{z=1}^{k} \alpha_z \right) \prod_{z=1}^{k} \frac{\pi(z)^{\alpha_z - 1}}{\Gamma(\alpha_z)} \tag{4.14}$$

This is a natural choice since the topics T_z are multinomial distributions, and Dirichlet is a conjugate prior for the multinomial. Despite its complexity, the LDA model has a relatively small number of free parameters: k parameters for the Dirichlet density and $|\mathcal{V}|-1$ probabilities for each of the k topic models T_z. The total number of parameters is $k|\mathcal{V}|$, which is almost the same as $k|\mathcal{V}|-1$ for a simple mixture model with as many topics, and substantially smaller than $k(|\mathcal{V}|-1) + (k-1)|\mathcal{C}_{train}|$ for a similar PLSI model. The estimation of free parameters is done using variational inference methods; complete details are provided in [12, 13].

4.2.6 A brief summary

We have described five generative formalisms that can be used to model strings of text. We presented them in the order of increasing modeling power, and increasing complexity of parameter estimation. Two of these models (unigram and Dirichlet) can only model homogeneous collections of text, and do not offer much in terms of capturing possible dependencies between the words. The other three explicitly model a collection as a mixture of k topics. These models allow for various degrees of dependence between the words. More specifically, dependence between words a and b can be encoded by making both words prominent in some topic model T_z and not prominent in other topics. The LDA model in particular has proven to be quite effective in modeling dependencies in a wide array of applications.

4.2.7 Motivation for a new model

Any one of the above models could be adopted as a generative process associated with relevance. However, we are not going to do that. In our opinion all of these models make the same unreasonable assumption. All five models postulate that any collection of strings can be modeled by a finite set of k aspects or topics[1]. The assumption is easy to make because it is intuitively appealing. As humans, we like to categorize things, arrange them on shelves, assign labels to them. We are always lured towards generalization, tempted to say that this news story is about *sports* and that one talks about *politics*; this image is a *landscape* and that other one is a *portrait* mixed with some *abstract art*. Even the current thesis will in the end be categorized as something along the lines of [*computer science; information retrieval; formal models*]. Topics and topical hierarchies are appealing because they allow us to deal with the complexity of the surrounding world. However, there is absolutely no intrinsic reason for using them in generative models. The dependency structure of large collections rarely corresponds to any given set of topics or to any hierarchy. And then there is the eternal question of how many topics are appropriate for a given model. Over the years, researchers have proposed a wide array of formal and heuristic techniques for selecting k, but somehow the question never seems to go away. The reason is that it is incredibly hard even for humans to pick k, even if they are dealing with a very specific task and are highly experienced in this task. Ask a librarian how many Dewey decimal codes she uses in her section of the library, and you may get a specific number k. Then ask her if k is enough, if she ever wanted to

[1] $k = 1$ for the unigram and Dirichlet models.

add just one more for that odd set of books that didn't seem to fit well with everything else. Or, suppose we try to come up with a general set of categories for news reporting. This task has actually been carried out by the Linguistic Data Consortium as part of topic annotations for the TDT project [5]. The result is a set of 13 broad categories, ranging from *politics* to *scandals* to *natural disasters*. Among the thirteen, which category accounts for the largest number of reports? The largest category is called "miscellaneous", and it contains news that simply does not fit in any of the other categories. This is the reality of information. Topic labels and hierarchies give a terrific way to present information to a human user, but one should not expect magic from using them as a foundation for probabilistic models of real data.

Our generative model will not be based on any sort of latent topic or aspect structure. But before we introduce our ideas, we will summarize the existing models in terms of a common framework – the framework that will naturally lead to our own model.

4.3 A Common Framework for Generative Models

Recall that we started this chapter by saying that we are interested in modeling exchangeable sequences of words. According to De Finetti's theorem, the joint distribution of the variables will take the form of equation (3.1), which we repeat below for the reader's convenience:

$$P(w_1 \ldots w_n) = \int_\Theta \left\{ \prod_{i=1}^{n} P_{\mathcal{S}_i,\theta}(w_i) \right\} p(d\theta)$$

where $P_{\mathcal{S}_i,\theta}(w_i)$ is some distribution appropriate for the space $\mathcal{S}_i$, θ denotes the set of parameters for that distribution, and $p(d\theta)$ is a probability measure over all possible settings of those parameters.

It is easy to show that all five models we discussed are special cases of the above equation. After all, the parameter space Θ can be anything, even a single point, and we can force the individual probabilities $P_{\mathcal{S}_i,\theta}(w_i)$ to be as elaborate as we want. For instance, if we wanted to represent PLSI or LDA, each component $P_{\mathcal{S}_i,\theta}(w_i)$ would involve a summation over a set of topics T_z weighted by some distribution $\pi(z)$. In this section we will demonstrate that all five models can be encompassed by a much simpler equation:

$$P(w_1 \ldots w_n) = \int_{I\!P_\mathcal{V}} \left\{ \prod_{i=1}^{n} u(w_i) \right\} p(du) \tag{4.15}$$

To get equation (4.15) we restricted the parameter space Θ to be the vocabulary simplex $I\!P_\mathcal{V} = \{u \in [0,1]^\mathcal{V} : |u| = 1\}$. Recall that the simplex $I\!P_\mathcal{V}$ is simply the set of all possible distributions over our vocabulary. The second change we made was to take the distribution $P_{\mathcal{S}_i,\theta}(\cdot)$, which could stand for a rather complex formula, and replace it with the single unigram model $u(\cdot)$. The generative process that corresponds to equation (4.15) would look like this:

1. Pick a unigram model u according to the generative density $p(du)$
2. For $i = 1 \ldots n$: pick word w_i from the unigram model u with probability $u(w_i)$

In the remainder of this section, we will show that all five models are equivalent to the above generative process. They only differ in the way they allocate the generative density $p(du)$ to the regions of $I\!P_\mathcal{V}$. The equivalence is fairly evident for the first three models, but it is not immediately clear that PLSI and LDA also fall into the formalism. In each of the five cases we will provide a brief geometric interpretation of the model. Geometrical details become particularly fascinating for the simplicial mixture models.

4.3.1 Unigram

The case of a unigram model is particularly simple. In order to get equation (4.1), we force the generative density to be the Dirac delta function:

$$p_{uni}(du) = \begin{cases} 1 & \text{if } du = \{U\} \\ 0 & \text{otherwise} \end{cases} \tag{4.16}$$

The effect of equation (4.16) is that we place all the generative density on a single point $U \in I\!P_\mathcal{V}$. Strictly speaking, $p_{uni}(du)$ is not a *density* function, rather it is a probability *measure*. Accordingly, the integral

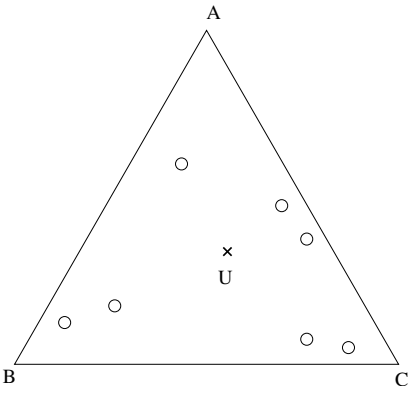


Figure 4.1. A geometric view of the unigram model over a three-word vocabulary. The triangle represents the probability simplex. Circles reflect empirical distributions of words in training strings. The point U marks the place where the generative density is non-zero.

in equation (4.15) is not the *Riemann* integral familiar to most engineers and scientists; it is the *Lebesgue* integral, which allows us to define and use all sorts of interesting probability measures. In the interest of accessibility to a wider audience, the remainder of this thesis will use the word *density* as a subsitute for *measure*. Similarly, all integrals from this point on will refer to *Lebesgue* integrals. A reader interested in the theory of Legesbue integration is invited to consult any graduate textbook on modern analysis.

The result of using the Dirac delta function as a generative density in equation (4.15) is as follows:

$$P(w_1 \ldots w_n) = \int_{I\!P_{\mathcal{V}}} \left\{ \prod_{i=1}^{n} u(w_i) \right\} p_{uni}(du) = \prod_{i=1}^{n} U(w_i) \tag{4.17}$$

Intuitively, the delta function simply picks out the necessary point $u = U$ and zeroes out the rest of the simplex $I\!P_{\mathcal{V}}$. We show a geometric view of the unigram model in Figure 4.1. We assume a toy vocabulary of three words $\{A, B, C\}$. The corresponding simplex is a triangle, each corner corresponds to a distribution that puts all the probability mass on a single word. The circles represent empirical distributions of words in training strings – the distributions we would get by counting how many times each word occurs in a given string, and then normalizing them so they add up to one. U marks our unigram urn – the point where all the generative density is concentrated on the simplex. The effect of using a unigram model is that we assume the single point U to be representative of all training strings (the circles).

4.3.2 Dirichlet

Showing the equivalence of a Dirichlet model is trivial: equations (4.15) and (4.6) are virtually identical. All we have to do is set the generative density $p_{dir}(du)$ to be the Dirichlet distribution with parameters α_v for $v \in \mathcal{V}$:

$$p_{dir}(du) = du \times \Gamma\left(\sum_{v \in \mathcal{V}} \alpha_v\right) \prod_{v \in \mathcal{V}} \frac{u(v)^{\alpha_v - 1}}{\Gamma(\alpha_v)} \tag{4.18}$$

We show a geometric view of the model in Figure 4.2. Everything is as before, only now instead of a single point U, the generative density is spread out over the entire simplex. Dashed circles represent possible isolines of the Dirichlet density. The mass would be concentrated around the center and would decrease towards the edges of the simplex.

4.3.3 Mixture

The mixture model consists of a set of topics $T_1 \ldots T_k$, and a weighting function $\pi(z)$ for mixing them together. In order to get our equation (4.15) to look like the mixture model, we will define the generative density as follows:

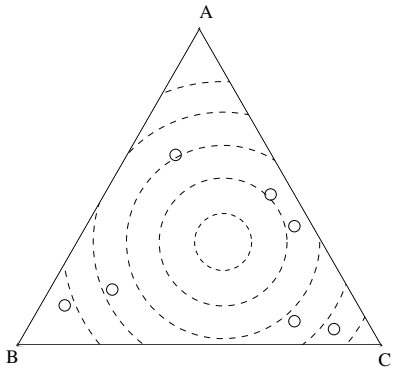


Figure 4.2. Geometric view of the Dirichlet model. Circles reflect training strings. Dashed lines represent a Dirichlet density over the three-word probability simplex.

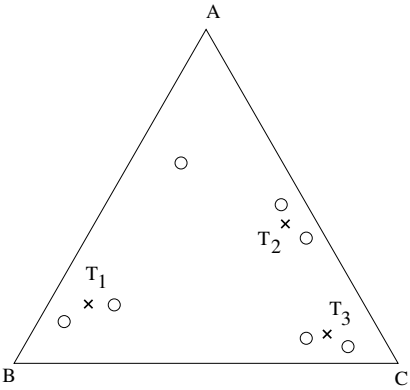


Figure 4.3. Geometric view of the mixture model. Circles reflect training strings. Generative density is concentrated over the points T_1, T_2 and T_3, which correspond to three topics in the mixture model.

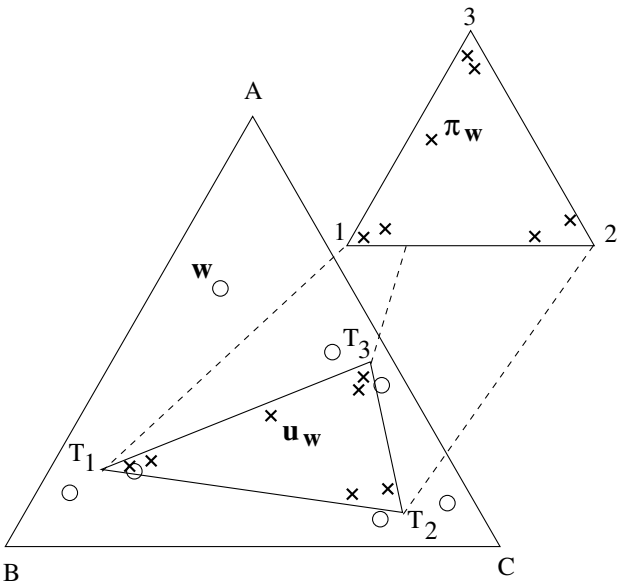


Figure 4.4. Geometric view of the PLSI model. To every training string $\mathbf{w}$ the model assigns a topic mixing distribution $\pi_{\mathbf{w}}$. In turn, this distribution defines to a single point $u_{\mathbf{w}}$ in the sub-simplex spanned by PLSI aspects T_1, T_2 and T_3. All the generative density in the model is concentrated over the points $u_{\mathbf{w}}$.

$$p_{mix}(du) = \begin{cases} \pi(z) & \text{if } du = \{T_z\} \text{ for some topic } z \\ 0 & \text{otherwise} \end{cases} \tag{4.19}$$

Now if we plug equation (4.19) into (4.15), we will get exactly equation (4.3), which defines the mixture model. The effect of density $p_{mix}(du)$ is to pick out k points from the simplex, each point corresponding to one of the topic model T_z. The points are weighted by $\pi(z)$. The integral over the simplex turns into a summation over the k points of $I\!P_{\mathcal{V}}$. Geometrically, the effect of using a mixture model is shown in Figure 4.3. The simplex and the training strings (circles) are unchanged. The generative density is concentrated on the three points marked as T_1, T_2 and T_3. Each point represents one of the k=3 topics in the mixture model. The topics seem to provide a good representation for a majority of the training strings, but note what happens when we have an outlier. The uppermost circle in the simplex corresponds to a training string that doesn't look like any of the other strings. The mixture model will force this string to be associated with T_2, the closest topic model. Naturally, T_2 will not be a good representation of that string. As we will see, this is a recurring problem with aspect representations: whenever we assume a fixed number of topics, the model will do a good job of capturing the bulk of the data, but will completely misrepresent any outliers in the collection.

4.3.4 PLSI

The Probabilistic LSI model is substantially more complex than our previous cases, so we will make extensive use of geometry to illustrate our argument. Recall that PLSI, like the mixture model, consists of a set of k topics T_z which are mixed on a word level using the distribution $\pi_{\mathbf{w}}(z)$. Let us try to figure out what PLSI looks like geometrically. In Figure 4.4 we have our usual word simplex $I\!P_{\mathcal{V}}$ with words A, B and C in the corners. As before, the training strings $\mathbf{w}$ are represented by small circles. We also have another simplex, with corners marked by 1, 2 and 3, it represents the set of all possible ways to mix k=3 components. PLSI endows every training string $\mathbf{w}$ with its own mixing distribution $\pi_{\mathbf{w}}(\cdot)$. These distributions are points in that smaller[2] simplex $I\!P_k$. Each corner of $I\!P_k$ corresponds to a single PLSI topic. For example, if some $\pi_{\mathbf{w}}$ were placed in corner 1, that would mean that string $\mathbf{w}$ is best modeled by the first topic alone.

Now we come to the key part of our argument. We observe that for each point in $I\!P_k$ there exists a corresponding point in the word simplex $I\!P_{\mathcal{V}}$. The corners $1 \ldots k$ of $I\!P_k$ correspond to the unigram topic

[2] In our example, there are three words in the vocabulary and three PLSI topics. In a realistic setting, the number of topics would be orders of magnitude smaller than vocabulary size.

models $T_1 \ldots T_k$, which are the points in $I\!P_{\mathcal{V}}$. Naturally, any linear mixture of topics $1 \ldots k$ will correspond to a similar mixture of unigram models $T_1 \ldots T_k$. These mixtures will form a *sub-simplex* $I\!P_T$ with unigram models $T_1 \ldots T_k$ as the corners. The important consequence of the mapping is this: for any mixing distribution $\pi_{\mathbf{w}}(\cdot)$ over the topics, there will be one and only one corresponding point $u_{\mathbf{w}}$ in the simplex $I\!P_{\mathcal{V}}$. The conclusion we draw from this is somewhat startling:

A generative version of PLSI is equivalent to a regular mixture model.

Indeed, we can take equation (4.11), which defines the probability that generative PLSI would assign to a string $w_1 \ldots w_n$, and re-write it as follows:

$$P_{lsi}(w_1 \ldots w_n) = \frac{1}{N} \sum_{\mathbf{w}} \prod_{i=1}^{n} \sum_{z=1}^{k} \pi_{\mathbf{w}}(z) T_z(w_i) = \frac{1}{N} \sum_{\mathbf{w}} \prod_{i=1}^{n} u_{\mathbf{w}}(w_i) \tag{4.20}$$

where $u_{\mathbf{w}}(\cdot)$ is a point in $I\!P_{\mathcal{V}}$ that corresponds to the mixing distribution $\pi_{\mathbf{w}}$:

$$u_{\mathbf{w}}(v) = \sum_{z=1}^{k} \pi_{\mathbf{w}}(z) T_z(v) \tag{4.21}$$

Why is this important? Because it gives us a much more realistic view of what PLSI really is. The universal view of PLSI is that it is a *simplicial*, or word-level, mixture with k aspects. We have just demonstrated that PLSI is a regular, string-level mixture with N components, one for each training string.[3] Another interesting observation is that all N components are restricted to the sub-simplex $I\!P_T$, formed by the original aspects. In a sense, PLSI defines a low-dimensional sandbox within the original parameter space $I\!P_{\mathcal{V}}$, and then picks N points from that sandbox to represent the strings in a collection. Since PLSI really has N mixture components, should we expect it to perform as a general, unrestricted mixture model with N topics? The answer is a resounding no: PLSI components are restricted by the k-aspect sandbox, they cannot travel to the regions of parameter space that contain outliers. PLSI will be able to capture more than a k-topic mixture, but nowhere near the unrestricted N-topic mixture.

Since PLSI turns out to be a restricted mixture, it is quite easy to define the generative density $p(du)$ that will make equation (4.15) equivalent to PLSI:

$$p_{lsi}(du) = \begin{cases} \frac{1}{N} & \text{if } du = \{u_{\mathbf{w}}\} \text{ for some training string } \mathbf{w} \\ 0 & \text{otherwise} \end{cases} \tag{4.22}$$

Here $u_{\mathbf{w}}$ is defined according to equation (4.21). It is easy to verify that plugging $p_{lsi}(du)$ into equation (4.15) will give us equation (4.20).

4.3.5 LDA

Our final case concerns the LDA model. The derivation is perhaps the hardest, but it will be greatly simplified by the results we proved in the previous section. Recall that LDA consists of a set of k topic models $\{T_z : z{=}1 \ldots k\}$, which are mixed on the word level using the mixing distribution $\pi(z)$. What makes LDA different from PLSI is that the mixing distribution π is not fixed but stochastically drawn from the simplex $I\!P_k$ under a Dirichlet prior $p_k(\pi)$. In Figure 4.5 we try to visualize LDA geometrically. The small triangle with corners 1, 2 and 3 represents the simplex $I\!P_k$. It is the set of all possible mixtures of $k{=}3$ components. Mixing distributions $\pi(\cdot)$ are elements of that set, and the dashed lines indicate a Dirichlet density $p_k(\cdot)$ over the set. The corners of $I\!P_k$ correspond to the topic models T_z, which are points in the vocabulary simplex $I\!P_{\mathcal{V}}$, the big triangle with corners A, B and C. The topic models $T_1 \ldots T_k$ make up the corners of a *sub-simplex* $I\!P_T$, a

[3] Our assertion refers to the marginalized (generative) version of PLSI. The original version of PLSI [58] is equivalent to a regular mixture with *only one* component, selected by an "oracle" mixture. In this way, the original PLSI is equivalent to the simple unigram model, it just gets to pick a perfect unigram urn for each new string.

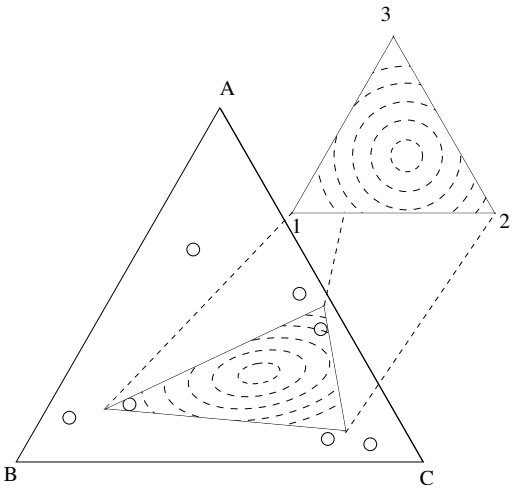


Figure 4.5. Geometric view of the LDA model. The generative density is restricted to the sub-simplex spanned by the k=3 LDA topics T_z inside the word simplex $A-B-C$. The density is induced by the Dirichlet prior over the mixing distributions $\pi(z)$, which are points in the smaller simplex $1-2-3$.

smaller triangle inside $\mathbb{P}_{\mathcal{V}}$. Every point u_π in the sub-simplex $\mathbb{P}_T$ corresponds to a mixing distribution $\pi(\cdot)$ in $\mathbb{P}_k$. Specifically, the point u_π is the result of mixing the unigram topic models $T_1 \ldots T_k$, weighted by $\pi(\cdot)$:

$$u_\pi(v) = \sum_{z=1}^{k} \pi(z) T_z(v) \tag{4.23}$$

The key observation to make is that Dirichlet density $p_k(\pi)$ over possible mixtures $\pi(\cdot)$ automatically induces a similar density $p_{lda}(u_\pi)$ over the unigram models u_π. This density is shown as dashed lines inside the sub-simplex $\mathbb{P}_k$. Assuming that topic models $T_1 \ldots T_k$ are not redundant[4], this new density can be expressed very simply:

$$p_{lda}(u_\pi) = p_k(\pi) \tag{4.24}$$

This density is defined for every point u_π in the sub-simplex $\mathbb{P}_T$. There are no holes because every point u_π corresponds to a unique mixing distribution $\pi \in \mathbb{P}_k$. However, the density $p_{lda}(u_\pi)$ is not defined for all points u in $\mathbb{P}_{\mathcal{V}}$, since some points in the larger simplex will not have a pre-image in $\mathbb{P}_k$. Fortunately, this is quite easy to fix. Since $\mathbb{P}_T$ is completely embedded inside $\mathbb{P}_{\mathcal{V}}$, we simply assign zero mass to any point $u \in \mathbb{P}_V V$ which happens to lie outside the sub-simplex $\mathbb{P}_T$. This gives us the following generative density:

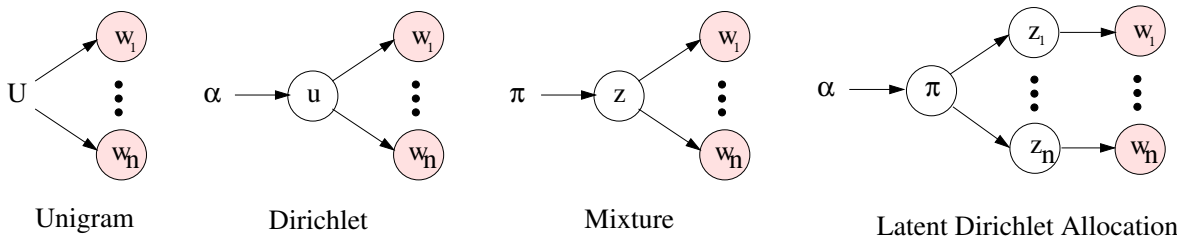
$$p_{lda}(du) = \begin{cases} \frac{p_k(\pi)}{V(\mathbb{P}_T)} \times du & \text{if } u = u_\pi \text{ for some mixing distribution } \pi(\cdot) \\ 0 & \text{otherwise} \end{cases} \tag{4.25}$$

Here u_π is given by equation (4.23), and $p_k(\pi)$ is the Dirichlet density over mixtures, defined in equation (4.14). $V(\mathbb{P}_T)$ represents the *volume* of the sub-simplex $\mathbb{P}_T$, dividing by it ensures that the generative density $p_{lda}(du)$ will integrate to one over the whole simplex $\mathbb{P}_{\mathcal{V}}$.

Let $f : \mathbb{P}_k \mapsto \mathbb{P}_T$ denote the function that takes a mixture distribution $\pi(\cdot)$ and produces the corresponding unigram model u_π according to equation (4.23). Since the topics $T_1 \ldots T_k$ are not redundant, f is a one-to-one mapping. Consequently, the inverse f^{-1} is well-defined: it takes a unigram model $u \in \mathbb{P}_T$ and maps it back to the mixture $\pi(\cdot)$ that produced u. Now let's take the generative density $p_{lda}(du)$, plug it into equation (4.15) and verify that we get the LDA model:

$$P(w_1 \ldots w_n) \quad = \quad \int_{\mathbb{P}_{\mathcal{V}}} \left\{ \prod_{i=1}^{n} u(w_i) \right\} p_{lda}(du)$$

[4]Topic models $T_1 \ldots T_k$ are redundant if they form a polytope of dimension smaller than $k-1$. Equivalently, topic models are redundant if two different mixing distributions $\pi_1 \neq \pi_2$ can produce the same mixture model: $\sum_z \pi_1(z) T_z = \sum_z \pi_2(z) T_z$. This is theoretically possible, but almost never happens in practice because k is orders of magnitude smaller than the dimensionality of $\mathbb{P}_{\mathcal{V}}$ (number of words in the vocabulary).



Unigram Dirichlet Mixture Latent Dirichlet Allocation

Figure 4.6. Graphical dependence networks corresponding to unigram, Dirichlet, Mixture model and LDA. Observable variables (words w_i) are shaded. Unshaded circles represent hidden variables. Labels without circles reflect priors. Arrows mark dependencies.

$$
\begin{aligned}
&= \int_{f(\mathbb{P}_k)} \left\{ \prod_{i=1}^{n} f(f^{-1}(u))(w_i) \right\} \frac{p_k(f^{-1}(u))}{V(\mathbb{P}_T)} du \\
&= \int_{\mathbb{P}_k} \left\{ \prod_{i=1}^{n} f(\pi)(w_i) \right\} p_k(\pi) d\pi \\
&= \int_{\mathbb{P}_k} \left\{ \prod_{i=1}^{n} \sum_{z=1}^{k} \pi(z) T_z(w_i) \right\} p_k(\pi) d\pi \qquad (4.26)
\end{aligned}
$$

Equation (4.26) leads us to the following conclusion:

LDA is equivalent to a Dirichlet model restricted to the topic sub-simplex.

The conclusion is in line with our result on PLSI: it appears that both models are equivalent to regular mixtures. There appears to be nothing special about the fact that these models mix the topics on a word level – it is equivalent to string-level mixing. Furthermore, both PLSI and LDA restrict the generative density to the same $k-1$ dimensional sandbox bounded by the aspects $T_1 \ldots T_k$. The main difference is that PLSI concentrates the density of N discrete points, while LDA spreads it out over the entire sandbox.

4.3.6 A note on graphical models

In the previous sections we demonstrated that equation (4.15) captures the following five generative models: unigram, Dirichlet, mixture, Probabilistic Latent Semantic Indexing and Latent Dirichlet Allocation. These five models represent the most dominant frameworks for dealing with exchangeable sequences. The last two models: PLSI, and especially LDA are widely regarded to be the most effective ways for capturing topical content, precisely because they allow aspects to be mixed at the word level of a string.

On the surface, all five models look very different from each other. For example, LDA involves many more variables than a simple unigram models, and dependencies between these variables are carefully crafted into a very specific structure. As an illustration, Figure 4.6 shows the graphical diagrams corresponding of the unigram model, the Dirichlet model, the Mixture model and the Latent Dirichlet Allocation model. The graphical networks are intuitive, they give a clear picture of dependencies in each model, and are well-suited for constructing models that are easy to interpret. For example, in the LDA network we see that there is a set of topics (z), and we are free to pick a different topic for every word w_i.

While we certainly do not want to argue about the usefullness of graphical models, we would like to point out that they are somewhat misleading. Figure 4.6 alone does not make it obvious that the Dirichlet model cannot represent multiple topics. In fact the Dirichlet graphical network looks *exactly* like the mixture model network, even though the two models have very different behavior. Similarly, by looking at the networks we cannot see that the only difference between LDA and the Dirichlet is that the former restricts generative density to the topic sub-simplex.

In our opinion, the geometric analysis of the models is much more illuminating. It allowed us to place the different models into the same framework and focus on what really makes the models behave differently: the way they allocate generative density to the unigram points u. We believe that focusing *directly* on density

allocation will ultimately lead to more powerful predictive models. We admit that these new models may not be nearly as intuitive as topic-based models such as PLSI and LDA, but as we mentioned before, we will opt for predictive power over elegance.

4.4 Kernel-based Allocation of Generative Density

We will now build upon the main result of the previous section. As we demonstrated, a large class of generative models can be represented by equation (4.15), which we re-state below for reading convenience:

$$P(w_1 \ldots w_n) = \int_{I\!P_{\mathcal{V}}} \left\{ \prod_{i=1}^{n} u(w_i) \right\} p(du)$$

The main difference between the models is not their dependence structure, but the way in which they allocate the density $p(du)$ to various regions of $I\!P_{\mathcal{V}}$. In this section we will construct two generative models by suggesting two possible choices for $p(du)$. We will not discuss the graphical structure of our models, focusing instead on their geometric representations, which we believe to be more expressive.

We have a collection of training strings $\mathcal{C}_{train}$, and would like to construct a generative model for it. We know that our generative model will take the form of equation (4.15), so instead of using maximum likelihood or Bayesian approaches, we will induce the model by *directly* constructing the density function $p(du)$. There exist a wide number of approaches for constructing a density function from a training collection. When we are faced with high-dimensional spaces and a small number of training examples[5], one of the best density estimation techniques is the *method of kernels*. An excellent exposition of the method can be found in [121]. The idea is to associate a *kernel* $K_{\mathbf{w}}(\cdot)$ with every training point $\mathbf{w}$. A kernel is a non-negative function, it is usually symmetric about zero, and must integrate to 1 over the space for which we are constructing our density. A kernel-based density estimate takes the form:

$$p_{ker}(du) = \frac{1}{N} \sum_{\mathbf{w}} K_{\mathbf{w}}(du) \tag{4.27}$$

The summation is performed over all $\mathbf{w} \in \mathcal{C}_{train}$. In a sense, a kernel density estimate is a mixture of a bunch of mini-densities, one associated with each training point $\mathbf{w}$. Kernel estimates are surprisingly powerful, especially in the face of limited examples. Perhaps most importantly, they have no tendency to ignore the outliers. Since each training point $\mathbf{w}$ has a kernel dedicated entirely to it, we are guaranteed that every outlier will have some probability mass floating around it. In the following two sections we will propose two possible ways of selecting a kernel $K_{\mathbf{w}}$ for training strings $\mathbf{w}$.

4.4.1 Delta kernel

Recall that whenever we presented a geometric interpretation of a generative model, we always had a set of small circles scattered over the word simplex $I\!P_{\mathcal{V}}$. Each of these circles corresponds to an empirical distribution $u_{\mathbf{w}}(\cdot)$ of words in that string. To get $u_{\mathbf{w}}(v)$ we simply count how many times each word v occurred in the string $\mathbf{w}$, and then normalize the counts to sum to one. The main problem with empirical distributions is that they contain many zeros. The vast majority of words will not occur in any given string. For example, a typical document in TREC contains no more than a thousand distinct words, whereas the total vocabulary size is 2 or 3 orders of magnitude larger. This means that empirical distributions $u_{\mathbf{w}}$ will concentrate around the edges and corners of the generative simplex $I\!P_{\mathcal{V}}$. Points on the edges turn out to be somewhat problematic,

[5] In most text collections, number of dimensions (unique words) is larger than the number of documents.

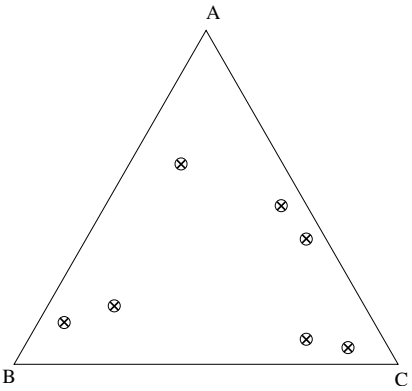


Figure 4.7. Density allocation with delta-kernels. The generative mass is concentrated over N points corresponding to empirical distributions of the N training strings.

so we will assume that empirical distributions $u_{\mathbf{w}}(\cdot)$ are smoothed out with background word probabilities $U(w)$:

$$u_{\mathbf{w}}(v) = \lambda \frac{n(v, \mathbf{w})}{|\mathbf{w}|} + (1 - \lambda) U(v) \tag{4.28}$$

The background probabilities $U(v)$ come directly from equation (4.1), with λ as the parameter that controls the degree of smoothing. The effect is to pull the points $u_{\mathbf{w}}$ from the edges of the simplex towards the point U in the middle. Now we can define our first kernel $K_{\mathbf{w}}$:

$$K_{\delta,\mathbf{w}}(du) = \begin{cases} 1 & \text{if } du = \{u_{\mathbf{w}}\} \\ 0 & \text{otherwise} \end{cases} \tag{4.29}$$

The kernel $K_{\delta,\mathbf{w}}$ is a delta function – it places all its probability mass on a single point $u_{\mathbf{w}}$. The corresponding density estimate $p_{dir}(du)$ is a set of N spikes, one for each training string $\mathbf{w}$. Each spike carries equal probability mass $\frac{1}{N}$, where N is the number of training strings. Figure 4.7 gives a geometric view of the model. The details are simple: the generative mass, denoted by crosses, will be concentrated on a set of N points, each corresponding to an empirical distribution $u_{\mathbf{w}}$ of some training string $\mathbf{w}$. A generative model based on Figure 4.7 will assign the following probability to a new string $w_1 \ldots w_n$:

$$\begin{aligned} P(w_1 \ldots w_n) &= \int_{I\!P_{\mathcal{V}}} \left\{ \prod_{i=1}^{n} u(w_i) \right\} \left\{ \frac{1}{N} \sum_{\mathbf{w}} K_{\delta,\mathbf{w}}(du) \right\} \\ &= \frac{1}{N} \sum_{\mathbf{w}} \int_{I\!P_{\mathcal{V}}} \left\{ \prod_{i=1}^{n} u(w_i) \right\} K_{\delta,\mathbf{w}}(du) \\ &= \frac{1}{N} \sum_{\mathbf{w}} \prod_{i=1}^{n} u_{\mathbf{w}}(w_i) \end{aligned} \tag{4.30}$$

Keep in mind that since section (4.3.1) we have been using the word *density* to refer to *measures*, and that the integral $\int_{I\!P_{\mathcal{V}}}$ is the *Lebesgue* integral.

4.4.1.1 Training the model

One should immediately recognize equation (4.30) as a special case of the mixture model with N components (eqiation 4.3). However, there is one very important difference that should not go unmentioned. A general mixture model with N topics has $N \times |\mathcal{V}|$ free parameters: we must estimate $|\mathcal{V}| - 1$ word probabilities for each of the N topic models, and then $N - 1$ additional parameters for the mixing distribution $\pi(z)$. The model given by equation (4.30) has *one* free parameter: the constant λ, which determines the degree of smoothing for empirical distributions $u_{\mathbf{w}}$. There are no other free parameters, since the empirical distributions $u_{\mathbf{w}}(v)$

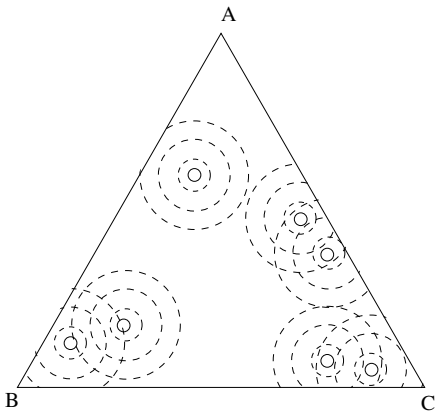


Figure 4.8. Density allocation with Dirichlet kernels. Each training example induces a Dirichlet bump in the simplex $A-B-C$. The bumps (shown with dashed lines) overlap to create some overall generative density $p(du)$ over the simplex.

are unambiguously specified by the training set. As a result, training the model is extremely easy and fast: all we have to do is find the value of λ that optimizes our favorite criterion on a held-out portion of the training set. One may counter that we have simply moved computational complexity from the training to the testing phase of the model. There is a grain of truth to that: once we have the value of λ, computing equation (4.30) is expensive, since it involves a summation over every training example $\mathbf{w}$. However, in practice we can apply a number of algorithmic optimizations that make the model quite fast. In our experience, the overall time required by the model (training + testing) is a small fraction of what is required by PLSI or LDA in a similar setting.

4.4.2 Dirichlet kernel

The delta kernel $K_{\delta,\mathbf{w}}$ presented in the previous section is a rather odd kind of kernel. In fact, using $K_{\delta,\mathbf{w}}$ is only reasonable because we applied it on top of a generative simplex. If we tried to apply the kernel directly to the space of observations we would be in a lot of trouble: by its very definition $K_{\delta,\mathbf{w}}$ can generate one and only one observation ($\mathbf{w}$), it cannot possibly generalize to new observations. In our case, the generalization ability was ensured by applying $K_{\delta,\mathbf{w}}$ to the space of distributions $I\!P_{\mathcal{V}}$. Every point $u(\cdot)$ in the interior of $I\!P_{\mathcal{V}}$ can generate any observation string $\mathbf{w}$, because it assigns a non-zero probability $u(v)$ to every word v. Zero probabilities happen only on the edges, which is precisely why we use equation (4.28) to pull our kernels away from the edges of the simplex. The problem with $K_{\delta,\mathbf{w}}$ is not its ability to generalize, but rather the relatively high variance of the resulting density estimator. Recall that with the delta kernel, our density estimate $p_{ker}(du)$ is a set of spikes, one for each training example. If we take a different training set $\mathcal{C}_{train}$, we will end up with a completely different set of spikes. One way to reduce the variance is to have each spike spread the mass in some region around itself – effectively turning spikes into a set of bumps. Geometrically, the resulting generative model would look like Figure 4.8. Each training example would generate a density around itself, and these densities would overlap to produce some kind of a bumpy surface over the simplex $A-B-C$. This surface will act as the generative density $p_{ker}(du)$. The variance with respect to different training sets will be much lower than for a set of spikes. We can achieve the geometry of Figure 4.8 by replacing the delta kernel $K_{\delta,\mathbf{w}}$ with a kernel based on the Dirichlet distribution:

$$K_{\alpha,\mathbf{w}}(du) = du \times \Gamma\left(\sum_{v\in\mathcal{V}} \alpha_{\mathbf{w}}(w)\right) \prod_{v\in\mathcal{V}} \frac{u(v)^{\alpha_{\mathbf{w}}(v)-1}}{\Gamma(\alpha_{\mathbf{w}}(v))} \tag{4.31}$$

We chose the Dirichlet distribution because it is a conjugate prior for the unigram models $u(\cdot)$. The position and width of each Dirichlet bump is determined by a set of parameters $\alpha_{\mathbf{w}}(v)$, one for each word in the

vocabulary. The parameters can be set in a number of ways, for instance:

$$\alpha_{\mathbf{w}}(v) \;=\; \lambda \cdot n(v, \mathbf{w}) \;+\; \sum_{\mathbf{x} \in \mathcal{C}_{train}} n(v, \mathbf{x}) \;+\; 1 \tag{4.32}$$

Here $n(v, \mathbf{w})$ gives the number of times we observe word v in string $\mathbf{w}$. The summation goes over all training strings $\mathbf{x}$, and λ is a constant that determines whether resulting Dirichlet bumps will be flat or spiky. Equation (4.32) looks like a heuristic, but in fact it can be derived as follows. We initialize the parameters by setting $\alpha_{\mathbf{w}}(v) = 1$, which gives us a uniform prior. Then we observe the entire training set $\mathcal{C}_{train}$. Next we observe λ copies of the string $\mathbf{w}$. The posterior expectation of $\alpha_{\mathbf{w}}(v)$ conditioned on these observations is exactly what we see in equation (4.32).

The benefit of using Dirichlet kernels comes to light when we try to express the overall likelihood of observing a new string $w_1 \ldots w_n$ from a generative model based on $K_{\alpha,\mathbf{w}}(du)$:

$$\begin{aligned} P(w_1 \ldots w_n) &= \int_{I\!P_{\mathcal{V}}} \left\{ \prod_{i=1}^{n} u(w_i) \right\} \left\{ \frac{1}{N} \sum_{\mathbf{w}} K_{\alpha,\mathbf{w}}(du) \right\} \\ &= \frac{1}{N} \sum_{\mathbf{w}} \frac{\Gamma(\sum_v \alpha_{\mathbf{w}}(v))}{\prod_v \Gamma(\alpha_{\mathbf{w}}(v))} \int_{I\!P_{\mathcal{V}}} \left\{ \prod_{i=1}^{n} u(w_i) \right\} \left\{ \prod_{v \in \mathcal{V}} u(v)^{\alpha_{\mathbf{w}}(v)-1} \right\} du \\ &= \frac{1}{N} \sum_{\mathbf{w}} \frac{\Gamma(\sum_v \alpha_{\mathbf{w}}(v))}{\Gamma(\sum_v \alpha_{\mathbf{w}}(v) + n)} \prod_{v \in \mathcal{V}} \frac{\Gamma(\alpha_{\mathbf{w}}(v) + n(v, w_1 \ldots w_n))}{\Gamma(\alpha_{\mathbf{w}}(v))} \end{aligned} \tag{4.33}$$

The good news stemming from equation (4.33) is that we do not have to perform numerical integration: $\int_{I\!P_{\mathcal{V}}}$ has a closed form solution solely because Dirichlet is a conjugate prior to the unigram. The bad news is that equation (4.33) is not susceptible to the same algorithmic trickery that allowed us to speed up equation (4.30). Accordingly, we have not been able to test equation (4.33) in large-scale retrieval scenarios. But in section 4.5 we will demonstrate that it constitutes an effective generative model, giving us a strong motivation for developing algorithms that will make equation (4.33) feasible.

4.4.2.1 Training the model

We observe that equation (4.33) involves just one free parameter: the constant λ which acts as inverse *bandwidth* of a kernel. Large values of λ yield kernels that are highly peaked, low values flatten out the bumps and also move them towards the center of the simplex. We can find the appropriate value of λ by performing a simple brute force search for the value that leads to highest likelihood for a held-out portion of the training set. Using a held-out subset is absolutely critical: if we do not, the model will quickly overfit, by driving λ towards positive infinity. This should come as no surprise: after all we are dealing with a kernel-based method. Curiously, very large values of λ will collapse the Dirichlet bumps into the delta spikes we discussed in the previous section. It will also drive the spikes towards the edges of the simplex $I\!P_{\mathcal{V}}$.

4.4.3 Advantages of kernel-based allocation

In the current section we proposed a kernel-based approach to distributing the probability mass over the generative simplex $I\!P_{\mathcal{V}}$. We believe this approach leads to novel generative models that have a number of distinct advantages over existing formulations. In this section we will briefly describe four of these advantages.

4.4.3.1 Handling rare events

Aspect-based models (PLSI, LDA) do an excellent job of capturing the bulk of the data in any collection. However, they are poorly suited for modeling outliers and rare events. Any outlier will automatically be folded into the topic structure, assigned to the closest aspect or mixture of aspects, even if these aspects really have nothing to do with the outlier string. Researchers have reported that aspect-based models show excellent performance on a large number of tasks, but most of these tasks are not affected by outlier events. These are tasks such as predictive modeling, clustering or coarse classification, where it is imperative to assign

reasonable likelihoods to the majority of strings. What happens to outliers really does not affect performance on these tasks: outlying events are so rare they have no chance of influencing the objective function.

However, there are other tasks where rare events are extremely important. In information retrieval, the user may be searching for a very specific subject, discussed by a very small set of documents. Or he may be interested in finding a rare piece of video footage. In Topic Detection and Tracking some of the most interesting events receive a very small amount of reporting. And for large TDT events it is essential to identify them as quickly as possible, when there are only one or two early reports on them. Aspect models are not suitable for these environments because they will tend to represent rare events as the mixture of prominent events. A kernel-based approach is much more appealing in this case. The outliers are not forced into a larger structure. Every rare event is guaranteed to have some small amount of probability mass around it.

The above argument does not mean that a kernel-based approach will only work for rare events. On the contrary, if a large number of examples fall into some region of $\mathbb{P}_{\mathcal{V}}$, their kernels will overlap and result in higher generative density over that region. For large events, kernel-based approach will behave in a way similar to the aspect model.

4.4.3.2 No structural assumptions

At its core, any topic model is a dimensionality reduction technique. The basic assumption in these models is that our observations are not as high-dimensional as they look. Usually we take every word to be a separate dimension, but words are not independent, they correlate in strange ways. Because of these correlations, we should not expect our collection to be a $|\mathcal{V}|$-dimensional spherical structure. It is more likely to be some peculiar manifold – a shape with intrinsic dimensionality much lower than $|\mathcal{V}|$. An aspect model like PLSI or LDA will try to "hug" this shape, approximate it by placing topics T_z as its vertices. In doing this, the aspect model will significantly reduce the dimensionality, without hurting the structure of the data. A smaller number of dimensions is very beneficial in a model because it means less sparseness, better estimates and less redundancy among the remaining dimensions.

The above reasoning is perfectly valid. We believe it is very plausible that the dimensionality of a text collection is not $\mathcal{V}$, that the data is embedded in some low-dimensional shape. But let us consider how mixture models approximate that shape. Any aspect model will approximate the shape by a *linear* polytope with topics placed in the corners. That may be perfectly correct, but it does constitute a strong assumption about the structure of our data. In other words, the shape may be low-dimensional, but why do we assume it is linear, or even convex? An approach based on kernels also makes an attempt to "hug" the shape enclosing the data, but it does the hugging in a very different manner. In a way it places a small sphere around each data point, and allows the sphere to touch and fuse with the neighboring spheres. The result is something like a thick ribbon winding through space. If we ignore the thickness, we get a low-dimensional manifold that tracks the data points. But this manifold does not have to be a polytope, as in the aspect model: there is no assumption of linear structure.

4.4.3.3 Easy to train

Kernel-based approaches are essentially non-parametric. As we described before, our methods have only one free parameter λ that controls the degree of estimator variance. One should not take this to mean that our model constitutes a compact representation with one parameter to store. Quite the opposite, a kernel-based model involves literally billions of parameters: we can easily have 100,000 training strings $\mathbf{w}$, over a vocabulary with over 100,000 words v. This would give us 10,000,000,000 Dirichlet parameters $\alpha_{\mathbf{w},v}$. But the important thing to realize is that these parameters are not free, they are completely determined by the data. We do not have to learn any of these parameters, we just need to create the infrastructure which allows us to store and access them efficiently. Once we have that infrastructure, "training" the model involves learning just one parameter λ, which is trivial. To contrast that, aspect models are fully parametric and routinely require inference for hundreds of thousands of parameters, e.g. 100 aspects with vocabulary trimmed to 1000 or so words.

4.4.3.4 Allows discriminative training

We would like to stress another important consequence of using a non-parametric approach. Even though we are talking about a generative model, the objective function we use for training λ does not have to be limited to likelihood. Since we have only one parameter the simple brute search will be just as fast as any other search method. This means that we don't have to stick to "nice" objective functions that facilitate gradient methods or EM. Instead, we can directly maximize the criterion that is appropriate for the problem at hand, regardless of whether that criterion is convex, differentiable or even continuous. Likelihood is the appropriate objective when we are specifically interested in the predictive qualities of a model. For ad-hoc retrieval we should maximize the average precision. For topic detection we should minimize the TDT cost function. To summarize: while our model is *generative*, it allows for a healthy bit of *discriminative* tuning.

4.5 Predictive Effectiveness of Kernel-based Allocation

We do not want to conclude this section without providing some empirical support for the alleged advantages of the proposed approach. In the present section we will briefly discuss a small scale experiment showing that kernel-based allocation does indeed lead to an effective generative model.

Consider a simple task of predictive text modeling. We have a collection of exchangeable strings $\mathcal{C}$. The collection will be partitioned into two subsets: the training set $\mathcal{A}$ and the testing set $\mathcal{B}$. Our goal is to use the training strings to estimate an effective generative model $P_{\mathcal{A}}(\mathbf{w})$. As we discussed in section 4.1.1, a model is effective if it does a good job of predicting the unseen testing portion $\mathcal{B}$ of our collection. *Predictive perplexity* is a widely accepted measure of predictive accuracy of a model; the measure is defined as:

$$\text{Perplexity}(\mathcal{B};\mathcal{A}) = \exp\left\{-\frac{1}{N_B}\sum_{\mathbf{w}\in\mathcal{B}}\log P_{\mathcal{A}}(\mathbf{w})\right\} \tag{4.34}$$

Here $P_{\mathcal{A}}(\mathbf{w})$ is the likelihood that our generative model will assign to a previously unseen string $\mathbf{w}$. The quantity inside the exponent is called the *entropy* of the data. The summation goes over all testing strings $\mathbf{w}$, and $N_{\mathcal{B}}$ represents the combined length of all testing strings. The logarithm typically has base 2, which enables interpreting the entropy in terms of *bits* of information.

Predictive perplexity has a very intuitive interpretation. A perplexity of K means that for each testing word the model is as *perplexed* as if it were choosing between K equally attractive options. For example, a perplexity of 1 means that there is no uncertainty at all, the model is absolutely sure of which word is going to come next. A perplexity of 2 means that the system is as uncertain as if it ruled out all but two words, but the remaining two were equally plausible. We would like to stress that with perplexity **lower means better**.

We will compare predictive perplexity of the following four models. **(i)** the unigram model (section 4.2.1), **(ii)** the Dirichlet model (section 4.2.3), **(iii)** density allocation with delta kernels (section 4.4.1) and **(iv)** density allocation with Dirichlet kernels (section 4.4.1). Our experiments will be done on two datasets. The first dataset contains 200 documents from the 1998 Associated Press corpus [80]. The documents contain 11,334 distinct words; average document length is 460. The second dataset contains 200 documents from the TDT-1 corpus [24]. TDT documents are slightly longer, average length is 540 words, but the number of distinct words is somewhat smaller: 9,379. In both cases we used the odd documents for estimating the model and even documents for testing.

The results are shown in Figure 4.9. We show testing-set perplexity as a function of the smoothing parameter λ, which is the only free parameter for all four models. The first observation we make from Figure 4.9 is that both kernel-based approaches lead to a dramatic reduction is predictive perplexity on both datasets. On the AP dataset, the perplexity is reduced by 20% from 2000-2100 to 1600-1700. On the TDT dataset we observe a more modest 14% reduction from around 1400 to around 1200. The figures demonstrate that kernel-based models are well suited for capturing whetever structure is present in the word occurrences.

The second interesting observation is that models based on Dirichlet distributions are more effective than models based on delta functions. Recall that Dirichlet distributions form "bumps" over the generative simplex, and Figure 4.9 suggests that these bumps are 5-7% more effective than the "spikes" corresponding to the unigram model and the δ-kernel model. There are two equally valid explanations for this behavior. The first is that delta spikes yield a high-variance density estimate over the generative simplex. The spikes

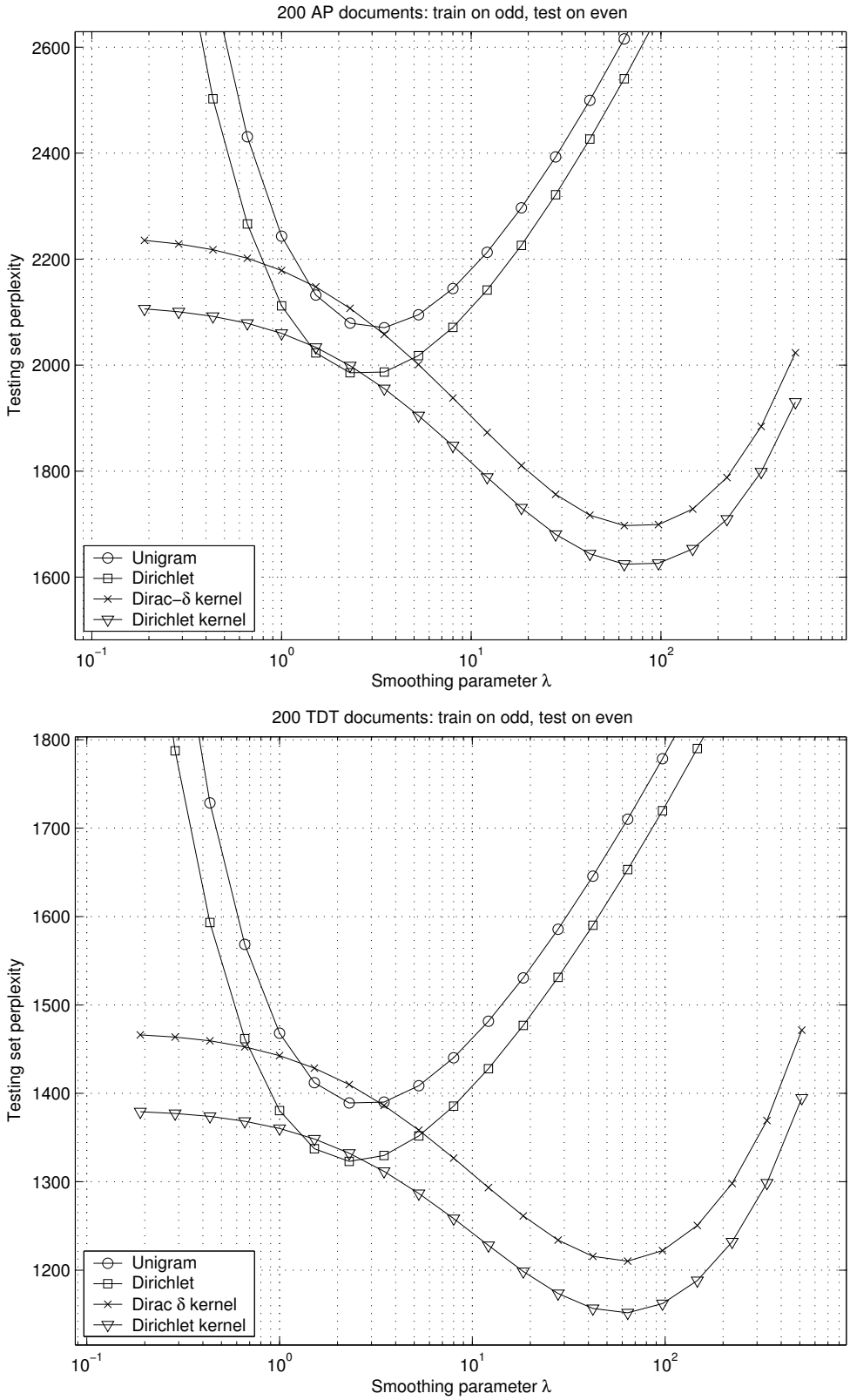


Figure 4.9. Predictive performance of generative models on the two datasets: AP (top) and TDT (bottom). Kernel-based models outperform the baselines by 14-20%. Dirichlet kernels yield better performance than delta kernels.

will jump around if we consider a different training subset $\mathcal{A}$. Dirichlet bumps, on the other hand, are much more stable. The peaks of these bumps may jump around, but we should not see drastic differences in the overall surface. The second explanation stems from our argument in the end of section 4.2.3. We showed that the main benefit of using a Dirichlet is that it will properly capture dependencies between successive occurrences of the same word. Word repetitions are quite common in natural language, so it should not come as a complete surprise that capturing repetitions leads to better predictive performance of a model.

The third, and final observation from Figure 4.9 is that the optimal setting of the smoothing parameter λ is quite stable across the two datasets. This suggests that we should be able to pick an optimal setting on one dataset, and use it to get good performance on a different dataset.

4.6 Summary

We began this chapter by arguing for the need to have powerful generative models for exchangeable sequences of data. The need was motivated by the generative relevance hypothesis. Recall that the GRH allows us to reduce the problem of retrieval to the problem of determining whether document d and query q could be samples from the same underlying generative model $P_{\mathcal{R}}(\cdot)$. Prompted by the GRH, we devoted this entire chapter to developing a powerful framework for generative models. Our development took the following steps:

1. We gave an overview of five existing generative models, starting with very basic formulas and progressing to advanced models, such as PLSI and LDA. We also argued that a limiting factor in these advanced models is the assumption of a fixed number of topics. While the assumption is very intuitive and appealing to humans, it carries no intrinsic benefit for generative modeling.

2. We demonstrated that all five models can be reduced to the following simple form:

$$P(w_1 \ldots w_n) = \int_{I\!P_{\mathcal{V}}} \prod_{i=1}^{n} u(w_i) p(du)$$

   We argued that the only difference between the models is in the way they allocate density $p(du)$ to regions of the generative simplex $I\!P_{\mathcal{V}}$. In the course of our arguments we made two surprising observations: **(i)** PLSI is a special case of a simple mixture model **(ii)** LDA is equivalent to a Dirichlet model restricted to the topic sub-simplex. We also argued that looking at geometric aspects of a generative model may be more elucidating than analyzing its graphical structure.

3. We proposed a new generative model, based on a kernel-based estimate of $p(du)$. We suggested two possible kernel types for the model: delta kernels (spikes) and Dirichlet kernels (bumps). We argued that the new model has the following attractive properties: **(i)** it makes no questionable assumptions about the structure of the space, **(ii)** it is able to handle rare events and preserve outliers, and **(iii)** it is reasonably efficient and allows for easy discriminative training when necessary.

4. We demonstrated that the new generative model is effective. The model yields a 14-20% reduction in predictive perplexity over two natural baselines.

This chapter completes the formal development of our model. We now have all the pieces necessary to construct operational systems based on the generative relevance hypothesis. As a relevance model, we will use the kernel-based density allocation model with delta kernels. In some cases computational complexity will force us to fall back to the simple Dirichlet model. The following chapter will describe specific instantiations of our relevance model in various search scenarios. The chapter will also contain extensive evaluation of retrieval effectiveness for each scenario.

CHAPTER 5

RETRIEVAL SCENARIOS

In the previous chapters we tried to keep our definitions very abstract – talking about conceptual "information" spaces, "transform" functions and unspecified "parameter" vectors. Our motivation was to keep the model as general as possible, so it would be applicable to a wide range of retrieval problems. Now is the time to bring our discussion down to earth and provide specific definitions for a number of popular retrieval scenarios. We will discuss the following seven retrieval scenarios:

1. **Ad-hoc retrieval:** we have a collection of English documents, and a short English query $\mathbf{q}$. The goal is to retrieve documents relevant to $\mathbf{q}$.
2. **Relevance feedback:** in addition to the query, the user provides us with a few examples of relevant documents. The goal is to retrieve more relevant documents.
3. **Cross-language retrieval:** we have a collection of Chinese documents and an English query. The goal is to find a Chinese relevant documents.
4. **Handwriting retrieval:** we have a set of historical manuscripts, represented as bitmap images. The goal is to search the collection using text queries.
5. **Image retrieval:** we have a collection of photographs. The goal is to identify photographs relevant to a given text query (e.g., find "tiger in the grass").
6. **Video retrieval:** we have a collection of video footage. The goal is to find video shots containing objects of interest (e.g., "forest fire").
7. **Topic detection and tracking:** we have a live stream of news reports. The goal is to organize the reports according to the events discussed in them.

For each of these retrieval scenarios we will address the following three issues:

(i) Definition. We will discuss the nature of a retrieval scenario, the goal we are trying to achieve, and the possible variations in the scenario.

(ii) Representation. We will provide a detailed description of how to adapt our model to the specific scenario. The representation section will answer the following questions:

- What is the appropriate "information" space $\mathcal{S}$?
- How do we represent documents and queries in that space?
- What will the relevance model $P_{\mathcal{R}}(\cdot)$ look like for this scenario?
- How can we estimate parameters of the relevance model?
- How do we rank information items (documents) in response to a query?

(iii) Experiments. We will describe a set of experiments comparing the accuracy generative relevance model against state-of-the-art baseline systems.

5.1 Ad-hoc Retrieval

We will start with the simplest possible scenario – ad-hoc retrieval. In this case we have a collection of documents $\mathcal{C}$, and a user's query $\mathbf{q}$. The documents in the collection have no structure and no meta-information. Aside from the query, we know nothing about the user and his information need. The query typically comes in the form of a small set of keywords, in the same language as the documents. In some cases, the user may provide a somewhat longer, syntactically well-formed request – a narrative discussing his information need. The goal of an information retrieval system is to rank all documents in the collection in such a way that relevant documents will end up at the top of the ranking. After the query is entered, no interaction is allowed to take place between the user and the system.

5.1.1 Representation

Since documents and queries share the same language, it makes sense to make that language the basis for our representation space $\mathcal{S}$. Let $\mathcal{V}$ be the vocabulary of our language and let M be an upper bound on document length. The representation space $\mathcal{S}$ will be the space of all possible sequences of length $M-1$, consisting of words from $\mathcal{V}$. The random variables $X_1 \ldots X_M$ will have the following meaning. X_1 will specify the target document length, a natural number less than M. Each subsequent variable $X_{i>1}$ will be a word from our vocabulary $\mathcal{V}$.

5.1.1.1 Document and query representation

The document generating transform D is very simple – it takes a representation $X = n, w_1 \ldots w_{M-1}$ and returns the first n words as the document:

$$D(X) = D(n, w_1 \ldots w_{M-1}) = n, w_1 \ldots w_n \tag{5.1}$$

The query transform Q is only slightly more complex. We assume that inside Q there is a heuristic that somehow decides for each word w_i whether it would make a good query word. Selected words form a subsequence $w_{i_1} \ldots w_{i_m}$, and Q returns that sub-sequence as the query:

$$Q(X) = Q(n, w_1 \ldots w_{M-1}) = m, w_{i_1} \ldots w_{i_m} \tag{5.2}$$

We do not have to bother with the details of that heuristic since we will always be dealing with already existing queries – for all intents and purposes that heuristic could be a deterministic human being. Since the representation space contains all possible strings over the vocabulary, we are guaranteed that D and Q can produce any document and any query.

5.1.1.2 Components of the relevance model

The first component of all representations is a document length n, or query length m. As a first approximation, we will assume that document / query length is independent of everything and follows a uniform distribution:

$$P_{\mathcal{R}}(X_1{=}n) = P_{\theta,\mathcal{S}_1}(n) = \frac{1}{M-1} \quad \text{for } n < M \tag{5.3}$$

All other variables $X_{i>1}$ take values in the same space: the vocabulary $\mathcal{V}$. A natural distribution over a vocabulary is a multinomial distribution. θ will be the set of parameters of that multinomial, i.e. for each word v, there will be a parameter θ_v which tells us how likely we are to observe the word v. For each $i > 1$ we have:

$$P_{\mathcal{R}}(X_i{=}v) = P_{\theta,\mathcal{S}_i}(v) = \theta_v \tag{5.4}$$

Since each θ is a distribution over the $\mathcal{V}$, the space Θ of all parameter settings will be the simplex of all possible probability distributions over the vocabulary: $\Theta = I\!P_{\mathcal{V}} = \left\{\vec{x}{\in}[0,1]^{\mathcal{V}} : |x| = 1\right\}$. We will adopt

kernel-based density allocation with delta kernels (equation 4.30). Under the above choices, the probability of observing a given query will take the following form:

$$\begin{aligned} P_{\mathcal{R}}(Q{=}\{m, q_1 \ldots q_m\}) &= \frac{1}{M-1} \int_{I\!P_{\mathcal{V}}} \left\{ \prod_{i=1}^{m} u(q_i) \right\} p_{ker,\delta}(du) \\ &= \frac{1}{N(M-1)} \sum_{\mathbf{d}} \prod_{i=1}^{m} u_{\mathbf{d}}(q_i) \end{aligned} \tag{5.5}$$

The summation goes over all training strings $\mathbf{d}$ which define the points where we place the kernels. In the ad-hoc scenario we have no special training collection, so we will use all available documents $\mathbf{d}{\in}\mathcal{C}$ as training strings. The probability of observing a given document has the same form as equation (5.5), except the product goes over the words $d_1 \ldots d_n$ in the document.

5.1.1.3 Document ranking

We will use both the document likelihood and the model comparison criteria for ranking documents. Recall that document likelihood, defined in section 3.2.3.1, was directly derived from Robertson's Probability Ranking Principle [110]. Model comparison, defined in section 3.2.3.3, is a generalization of query likelihood that allows for estimation both on the document and query side. As we discussed in section 3.2.3.4, both ranking criteria require us to compute the expected value of the parameter vector conditioned on the query observation. In our case, parameter vectors are unigram distributions $u(\cdot)$ over the vocabulary. Under the distribution given by equation (5.5) the expected parameter vector $E_q u(v)$ will take the form:

$$E_q u(v) = \frac{\int_{I\!P_{\mathcal{V}}} u(v) \left\{ \prod_{i=1}^{m} u(q_i) \right\} p_{ker,\delta}(du)}{\int_{I\!P_{\mathcal{V}}} \left\{ \prod_{i=1}^{m} u(q_i) \right\} p_{ker,\delta}(du)} = \frac{\sum_{\mathbf{d}} u_{\mathbf{d}}(v) \prod_{i=1}^{m} u_{\mathbf{d}}(q_i)}{\sum_{\mathbf{d}} \prod_{i=1}^{m} u_{\mathbf{d}}(q_i)} \tag{5.6}$$

As before, summation goes over all documents $\mathbf{d}$ in the collection, and the empirical distributions $u_{\mathbf{d}}(\cdot)$ are computed according to equation (4.28). Note that the N and $(M-1)$ cancel out. Under the document likelihood criterion we will be ranking the documents according to the ratio:

$$\frac{P(D{=}d_1 \ldots d_n | R{=}1)}{P(D{=}d_1 \ldots d_n | R{=}0)} \approx \frac{\prod_{i=1}^{n} E_q u(d_i)}{\prod_{i=1}^{n} E_0 u(d_i)} \tag{5.7}$$

Here $E_q u(\cdot)$ comes directly from equation (5.6), and $E_0 u(\cdot)$ is the background distribution over the words, computed by taking their relative frequency in the collection. As we argued before, the background distribution is going to provide a good reflection of word probabilities in the non-relevant documents ($R{=}0$). When we use model comparison, we will rank the documents d by the relative entropy between $E_q u(\cdot)$ and the empirical distribution $u_d(\cdot)$:

$$-D(E_q u || u_d) \propto \sum_{v \in \mathcal{V}} E_q u(v) \log u_d(v) \tag{5.8}$$

As before, $E_q u(\cdot)$ is given by equation (5.6), and $u_d(\cdot)$ is the empirical distribution of words in the document d, computed according to equation (4.28).

5.1.2 Experiments

In this section we turn our attention to evaluating the performance of the proposed retrieval model on a number of TREC collections. The experiments were originally reported in [73]. The remainder of this section is organized as follows. We start by describing the evaluation metrics and the corpora which were used in our experiments. In section 5.1.2.3 we will discuss the baseline systems that will be used to gauge the retrieval effectiveness of our model; the comparisons themselves are provided in section 5.1.2.4. In section 5.1.2.5 we compare the retrieval effectiveness of document-likelihood and query-likelihood ranking criteria.

Name	Sources	Years	#Docs	#Terms	dl	cf	Queries	ql
AP	Associated Press	89-90	242,918	315,539	273	210	51-150	4.32
FT	Financial Times	91-94	210,158	443,395	237	112	251-400	2.95
LA	Los Angeles Times	89-90	131,896	326,609	290	117	301-400	2.52
WSJ	Wall Street Journal	87-92	173,252	185,903	265	247	1-200	4.86
TDT2	AP, NYT, CNN, ABC, PRI, VOA	1998	62,596	122,189	167	85	TDT	3.02

Table 5.1. Information for the corpora used in ad-hoc retrieval experiments. *dl* denotes average document length, *cf* stands for average collection frequency of a word, and *ql* represents average number of words per query.

5.1.2.1 Evaluation Paradigm

The primary goal of any Information Retrieval system lies in identifying a set $\mathcal{C}_{\mathcal{R}} \subset \mathcal{C}$ of documents relevant to some information need. This is a set-based decision task, but in practice most retrieval systems are evaluated by how well they can *rank* the documents in the collection. Let $d_1, d_2, \ldots d_N$ denote some ordering of the documents in the collection. Then, for every rank k, we can compute *recall* as the number of relevant documents that were observed in the set $\{d_1 \ldots d_k\}$, divided by the total number of relevant documents in the collection. Similarly, *precision* is defined as the number of relevant documents among $\{d_1 \ldots d_k\}$ divided by k. System performance is evaluated by comparing precision at different levels of recall, either in a form of a table (e.g. Table 5.2), or as a recall-precision graph (e.g. Figure 5.1). A common objective is to increase precision at all levels of recall. For applications that require interaction with a user, it is common to report precision at specific ranks, e.g. after 5 or 10 retrieved documents. When one desires a single number as a measure of performance, a popular choice is *average precision* defined as an arithmetic average of precision at every rank where a relevant document occurs. Another possible choice is R-precision, precision that is achieved at rank R, where R is the number of relevant documents in the dataset. In all of these measures, precision values are usually averaged across a large set of queries with known relevant sets.

We will adopt average precision as the primary evaluation measure for all the experiments in this paper. In most cases we will also report precision at different recall levels and precision at specific ranks. When possible, we will report the results of statistical significance tests.

5.1.2.2 Datasets and processing

We use five different datasets in our evaluation of adhoc retrieval effectiveness. Table 5.1 provides detailed information for each dataset. All five datasets contain news releases; the majority of them are print media, although the TDT2 corpus contains a significant broadcast component. The datasets vary in size, time frame, and word statistics. All datasets except TDT2 are homogeneous, i.e. they contain documents from a single source. For each dataset there is an associated set of topics, along with human relevance judgments. For the TDT2 dataset the judgments are exhaustive, meaning that every document has been manually labeled as either relevant or non-relevant for every topic. The other four datasets contain *pooled* judgments, i.e. only top-ranked documents from a set of retrieval systems were judged with respect to each topic by annotators at NIST. TREC topics come in the form of queries, containing title, description and narrative portions. We used only the titles, resulting in queries which are 3-4 words in length. TDT topics are defined by a set of examples and do not have associated queries. However, short titles were assigned to them by annotators, and we used these titles as queries.

Prior to any experiments, each dataset was processed as follows. Both documents and queries were tokenized on whitespace and punctuation characters. Tokens with fewer than two characters were discarded. Tokens were then lower-cased and reduced to their root form by applying the Krovetz stemmer used in the InQuery engine [3]. The stemmer combines morphological rules with a large dictionary of special cases and exceptions. After stemming, 418 stop-words from the standard InQuery [3] stop-list were removed. All of the remaining tokens were used for indexing, and no other form of processing was used on either the queries or the documents.

5.1.2.3 Baseline systems

We will compare retrieval performance of generative relevance models (**RM**) against four established baselines:

tf.idf Our first baseline represents one of the most widely-used and successful approaches to adhoc retrieval. We use the InQuery [3] modification of the popular Okapi *BM25* [115] weighting scheme. Given a query $Q = q_1 \dots q_k$, the documents D are ranked by the following formula:

$$S(Q, D) = \sum_{q \in Q} \frac{n(q, D)}{n(q, D) + 0.5 + 1.5 \frac{dl}{avg.dl}} \frac{\log\left(0.5 + N/df(q)\right)}{\log\left(1.0 + \log N\right)} \tag{5.9}$$

Here $n(q, D)$ is the number of times query word q occurs in document D, dl is the length of document D, $df(q)$ denotes the number of documents containing word q, and N stands for the total number of documents in the collection. Equation (5.9) represents a heuristic extension of the classical probabilistic models of IR, discussed in section 2.3.2. The formula is remarkable for its consistently good performance in yearly TREC evaluations.

LCA From an IR standpoint, our retrieval model contains a massive query expansion component (see equation 5.6). To provide a fair comparison, we describe performance of a state-of-the-art heuristic query expansion technique: Local Context Analysis (LCA) [147]. LCA is a technique for adding highly-related words to the query, in the hope of handling synonymy and reducing ambiguity. Given a query $Q = q_1 \dots q_k$, and a set of retrieved documents R, LCA ranks all words w in the vocabulary by the following formula:

$$Bel(w; Q) = \prod_{q \in Q} \left(0.1 + \frac{1/\log n}{1/idf_w} \log \sum_{D \in R} D_q D_w \right)^{idf_q} \tag{5.10}$$

Here n is the size of the retrieved set, idf_w is the inverse document frequency [113] of the word w; D_q and D_w represent frequencies of words w and q in document D. To perform query expansion, we add m highest-ranking words to the original query $q_1 \dots q_k$, and perform retrieval using **tf.idf** method described above. n and m represent parameters that can be tuned to optimize performance. Based on preliminary experiments, we set $n = 10$ and $m = 20$.

LM Our third baseline represents the language-modeling framework, pioneered by Ponte and Croft [102] and further developed by a number of other researchers [123, 56, 87]. The framework was discussed in detail in section 2.3.3. Recall that in the language-modeling framework we view the query $Q = q_1 \dots q_k$ as an i.i.d. random sample from some unknown model representing a perfectly relevant document. Documents D are ranked by the probability that $q_1 \dots q_k$ would be observed during random sampling from the model of D:

$$P(Q|D) = \prod_{q \in Q} \left(\lambda_D \frac{n(q, D)}{dl} + (1 - \lambda_D) P(q) \right) \tag{5.11}$$

Here $n(q, D)$ is the number of times q occurs in D, dl is the length of D, and $P(q)$ is the background probability of the query word q, computed over the entire corpus. The smoothing parameter λ_D was set to the Dirichlet [153] estimate $\lambda_D = \frac{dl}{dl+\mu}$. Based on previous experiments, μ was set to 1000 for all TREC corpora and to 100 for the TDT2 corpus.

Recall from section 3.2.3.5 that this method of ranking is equivalent to using relative entropy where the relevance model was replaced with the empirical distribution of words in Q. Curiously, it is also equivalent to using our generative relevance model with a simple Dirichlet density (section 4.3.2) instead of a kernel-based density (section 4.4).

WSJ: TREC queries 1-200 (title)

	tf.idf	LCA	%chg	LM	%chg	LM+X	%chg	RM	%chg
Rel	20982	20982		20982		20982		20982	
Rret	11798	12384	4.97*	12025	1.92*	12766	8.20*	13845	17.35*
0.00	0.634	0.666	4.9	0.683	7.6*	0.653	3.0	0.686	8.1*
0.10	0.457	0.465	1.8	0.474	3.8	0.481	5.4	0.533	16.8*
0.20	0.383	0.380	-0.8	0.395	3.0	0.403	5.1	0.463	20.7*
0.30	0.334	0.333	-0.3	0.340	1.8	0.352	5.3	0.403	20.6*
0.40	0.287	0.283	-1.4	0.288	0.2	0.307	6.7	0.350	21.9*
0.50	0.240	0.240	0.1	0.246	2.7	0.270	12.6*	0.304	26.6*
0.60	0.191	0.195	2.3	0.203	6.5	0.226	18.5*	0.254	33.0*
0.70	0.138	0.153	11.1*	0.158	14.7*	0.178	29.0*	0.196	42.2*
0.80	0.088	0.108	22.8*	0.110	24.9*	0.133	50.7*	0.146	66.1*
0.90	0.049	0.061	25.3*	0.074	51.8*	0.080	64.2*	0.085	73.3*
1.00	0.011	0.013	25.4	0.017	55.6*	0.022	104.3*	0.022	99.7*
Avg	0.238	0.244	2.89	0.253	6.61*	0.265	11.46*	0.301	26.51*
5	0.4360	0.4470	2.5	0.4480	2.8	0.4620	6.0*	0.5170	18.6*
10	0.4060	0.4185	3.1	0.4140	2.0	0.4220	3.9	0.4675	15.1*
15	0.3830	0.3827	-0.1	0.3900	1.8	0.3963	3.5	0.4463	16.5*
20	0.3662	0.3580	-2.3	0.3737	2.0	0.3805	3.9	0.4258	16.2*
30	0.3390	0.3277	-3.3	0.3498	3.2*	0.3483	2.8	0.3933	16.0*
100	0.2381	0.2318	-2.7	0.2432	2.1*	0.2513	5.5	0.2744	15.3*
200	0.1725	0.1713	-0.7	0.1760	2.1*	0.1822	5.6*	0.1996	15.7*
500	0.0978	0.1018	4.1*	0.1009	3.1*	0.1041	6.4*	0.1138	16.3*
1000	0.0590	0.0619	5.0*	0.0601	1.9*	0.0638	8.2*	0.0692	17.4*
RPr	0.2759	0.2753	-0.23	0.2835	2.73*	0.2871	4.03	0.3162	14.58*

Table 5.2. Comparison of Relevance Models (**RM**) to the baseline systems: (**tf.idf**) Okapi / InQuery weighted sum, (**LCA**) tf.idf with Local Context Analysis, (**LM**) language model with Dirichlet smoothing, (**LM+X**) language model with Ponte expansion. Relevance Models noticeably outperform all baseline systems. Stars indicate statistically significant differences in performance with a 95% confidence according to the Wilcoxon test. Significance tests are performed against the **tf.idf** baseline.

LM+X The language-modeling framework does not contain an inherent query expansion component. In his thesis, Ponte [101] developed a heuristic query expansion approach that demonstrated respectable performance when combined with the ranking formula described above. Given the set of documents R, retrieved in response to the original query, we rank all the words w by the following formula:

$$Bel(w;R) = \sum_{D \in R} log\left(\frac{P(w|D)}{P(w)}\right) \tag{5.12}$$

Here $P(w|D)$ is the smoothed relative frequency of word w in D, and $P(w)$ is the background probability of w. m highest-ranking words are added to the original query $q_1 \ldots q_k$, and retrieval was performed according to the **LM** method described above. Based on prior experiments, we added m=5 words from the n=5 top-ranked documents.

5.1.2.4 Comparing the generative relevance model to baselines

Table 5.2 presents the performance of five systems on the Wall-Street Journal (WSJ) dataset with TREC title queries 1 - 200. We observe that in general, language-modeling approaches (**LM** and **LM+X**) are slightly superior to their heuristic counterparts. Query expansion techniques lead to significant improvements at high recall. Relevance models (**RM**) noticeably outperform all four baselines at all levels of recall, and also in terms of average precision and R-precision (precision at the number of relevant documents). Improvements

	AP: TREC queries 51-150 (title)					FT: TREC queries 251-400 (title)				
	tf.idf	LM	%chg	RM	%chg	tf.idf	LM	%chg	RM	%chg
Rel	21809	21809		21809		4816	4816		4816	
Ret	10115	10137	0.2	12525	23.8*	2541	2593	2.0*	3197	25.8*
0.0	0.644	0.643	-0.1	0.632	-1.8	0.531	0.561	5.5	0.535	0.8
0.1	0.442	0.436	-1.4	0.484	9.4*	0.415	0.421	1.4	0.430	3.7
0.2	0.359	0.349	-2.8	0.425	18.4*	0.353	0.355	0.4	0.368	4.4
0.3	0.308	0.299	-3.0*	0.379	23.1*	0.291	0.303	4.3	0.316	8.8*
0.4	0.255	0.246	-3.5	0.333	30.8*	0.249	0.258	3.6	0.282	13.3*
0.5	0.212	0.209	-1.4	0.289	35.9*	0.213	0.230	7.8	0.256	20.0*
0.6	0.176	0.170	-3.6*	0.246	39.8*	0.158	0.187	17.9*	0.210	32.6*
0.7	0.128	0.130	1.3	0.184	43.9*	0.108	0.137	26.4*	0.160	47.3*
0.8	0.084	0.086	2.8	0.128	52.0*	0.078	0.102	30.6*	0.128	63.3*
0.9	0.042	0.048	14.9	0.071	70.8*	0.058	0.078	32.9*	0.089	53.0*
1.0	0.016	0.022	38.8*	0.018	10.5	0.042	0.066	56.5*	0.059	40.1*
Avg	0.222	0.219	-1.4	0.277	25.0*	0.211	0.230	8.8*	0.246	16.7*
5	0.430	0.457	6.1	0.481	11.7*	0.306	0.332	8.5	0.331	8.0
10	0.420	0.434	3.4	0.470	11.8*	0.263	0.276	4.9	0.283	7.7
15	0.410	0.417	1.6	0.457	11.3*	0.224	0.251	12.2*	0.244	9.0*
20	0.396	0.409	3.4	0.448	13.3*	0.201	0.225	12.0*	0.218	8.8*
30	0.380	0.390	2.7	0.429	12.9*	0.170	0.187	10.2*	0.190	11.7*
100	0.302	0.305	0.8	0.354	17.0*	0.095	0.099	4.4*	0.111	17.0*
200	0.240	0.242	1.2	0.294	22.6*	0.061	0.064	5.2*	0.074	21.7*
500	0.156	0.157	0.9	0.193	23.7*	0.031	0.032	1.6*	0.039	24.9*
1000	0.102	0.102	0.2	0.127	23.8*	0.018	0.019	2.0*	0.023	25.8*
RPr	0.272	0.268	-1.6	0.315	15.7*	0.227	0.236	3.8*	0.236	4.0

Table 5.3. Comparison of Relevance Models (**RM**) to the InQuery (**tf.idf**) and language-modeling (**LM**) systems. Relevance Model significantly outperforms both baselines. Stars indicate statistically significant differences in performance with a 95% confidence according to the Wilcoxon test. Significance is against the **tf.idf** baseline.

over **tf.idf** are all statistically significant at the 95% confidence level according to the Wilcoxon signed-rank test [144].

In Tables 5.3 and 5.4 we show the performance on the remaining four datasets: AP, FT, LA and TDT2. We compare relevance models (**RM**) with two of the four baselines: **tf.idf** and **LM**. As before, we notice that language modeling approach (**LM**) is somewhat better than the heuristic **tf.idf** ranking. However, the improvements are not always consistent, and rarely significant. Relevance models demonstrate consistent improvements over both baselines on all four datasets. Compared to **tf.idf**, overall recall of relevance models is higher by 12% - 25%, and average precision is up by 15% - 25%. Improvements are statistically significant with 95% confidence according to the Wilcoxon test.

5.1.2.5 Comparing different document ranking criteria

In section 3.2.3 we discussed two ways of ranking documents in response to the query: document likelihood and query likelihood. We also discussed how both ways can be generalized to model comparison. In section 3.2.3.5 we argued that query likelihood is potentially superior to document likelihood, even if the relevance model can be estimated perfectly. In this section we provide empirical support for our arguments.

In our comparison, we want to factor out the effects of what method we use for estimating the relevance model. We start by estimating the model $E_{\mathbf{q}}u$ from a set of 1, 5 or 10 training documents. In each case we use smoothed maximum-likelihood models as described in section. Smoothing parameters were tuned individually to maximize performance. The quality of the resulting rankings are shown in the top portion of Figure 5.1. Solid lines reflect the performance of rankings based on query likelihood (cross-entropy), dashed lines correspond to the document likelihood (probability ratio). We observe two consistent effects. First,

	LA: TREC queries 301-400 (title)					TDT2: TDT topics 1-100 (title)				
	tf.idf	LM	%chg	RM	%chg	tf.idf	LM	%chg	RM	%chg
Rel	2350	2350		2350		7994	7994		7994	
Ret	1581	1626	2.9	1838	16.3*	5770	5399	-6.4	6472	12.2*
0.0	0.586	0.619	5.7	0.566	-3.5	0.846	0.843	-0.4	0.854	0.9
0.1	0.450	0.486	8.2*	0.474	5.5	0.794	0.797	0.4	0.831	4.5*
0.2	0.356	0.362	1.9	0.384	8.1	0.755	0.748	-0.9	0.806	6.8*
0.3	0.295	0.316	7.0*	0.332	12.5*	0.711	0.705	-0.9	0.785	10.3*
0.4	0.247	0.273	10.4	0.286	15.6*	0.663	0.669	0.8	0.766	15.5*
0.5	0.217	0.238	9.8	0.264	21.7*	0.614	0.616	0.3	0.742	20.9*
0.6	0.164	0.197	19.7	0.206	25.5*	0.565	0.563	-0.5	0.704	24.6*
0.7	0.129	0.159	23.0*	0.174	34.9*	0.528	0.517	-2.1	0.675	27.8*
0.8	0.100	0.123	23.1	0.138	37.5*	0.485	0.477	-1.5*	0.648	33.6*
0.9	0.050	0.074	47.0	0.085	69.4*	0.397	0.394	-0.7*	0.587	48.0*
1.0	0.042	0.059	41.5*	0.056	32.6*	0.297	0.307	-3.5	0.477	60.4*
Avg	0.223	0.247	10.6*	0.258	15.6*	0.596	0.592	-0.6	0.709	18.9*
5	0.331	0.337	1.9	0.357	8.0	0.590	0.598	1.4	0.671	13.8*
10	0.274	0.282	3.0	0.299	9.3	0.538	0.543	0.8	0.607	12.8*
15	0.237	0.250	5.7	0.262	10.6	0.485	0.502	3.6*	0.556	14.6*
20	0.213	0.220	3.4	0.228	7.2	0.453	0.457	0.9	0.508	12.2*
30	0.179	0.186	4.0	0.195	8.7	0.402	0.398	-0.9	0.453	12.8*
100	0.092	0.094	2.3	0.102	11.8	0.247	0.243	-1.1	0.288	17.0*
200	0.054	0.057	5.2	0.066	20.2*	0.169	0.165	-2.4*	0.193	14.2*
500	0.028	0.029	5.9	0.033	18.9*	0.095	0.092	-3.8	0.109	14.4*
1000	0.016	0.017	2.8	0.019	16.3*	0.060	0.056	-6.4	0.067	12.2*
RPr	0.235	0.257	9.1*	0.251	6.5	0.564	0.567	0.51	0.672	19.3*

Table 5.4. Comparison of Relevance Models (**RM**) to the InQuery (**tf.idf**) and language-modeling (**LM**) systems. Relevance Model significantly outperforms both baselines. Stars indicate statistically significant differences in performance with a 95% confidence according to the Wilcoxon test. Significance is against the **tf.idf** baseline.

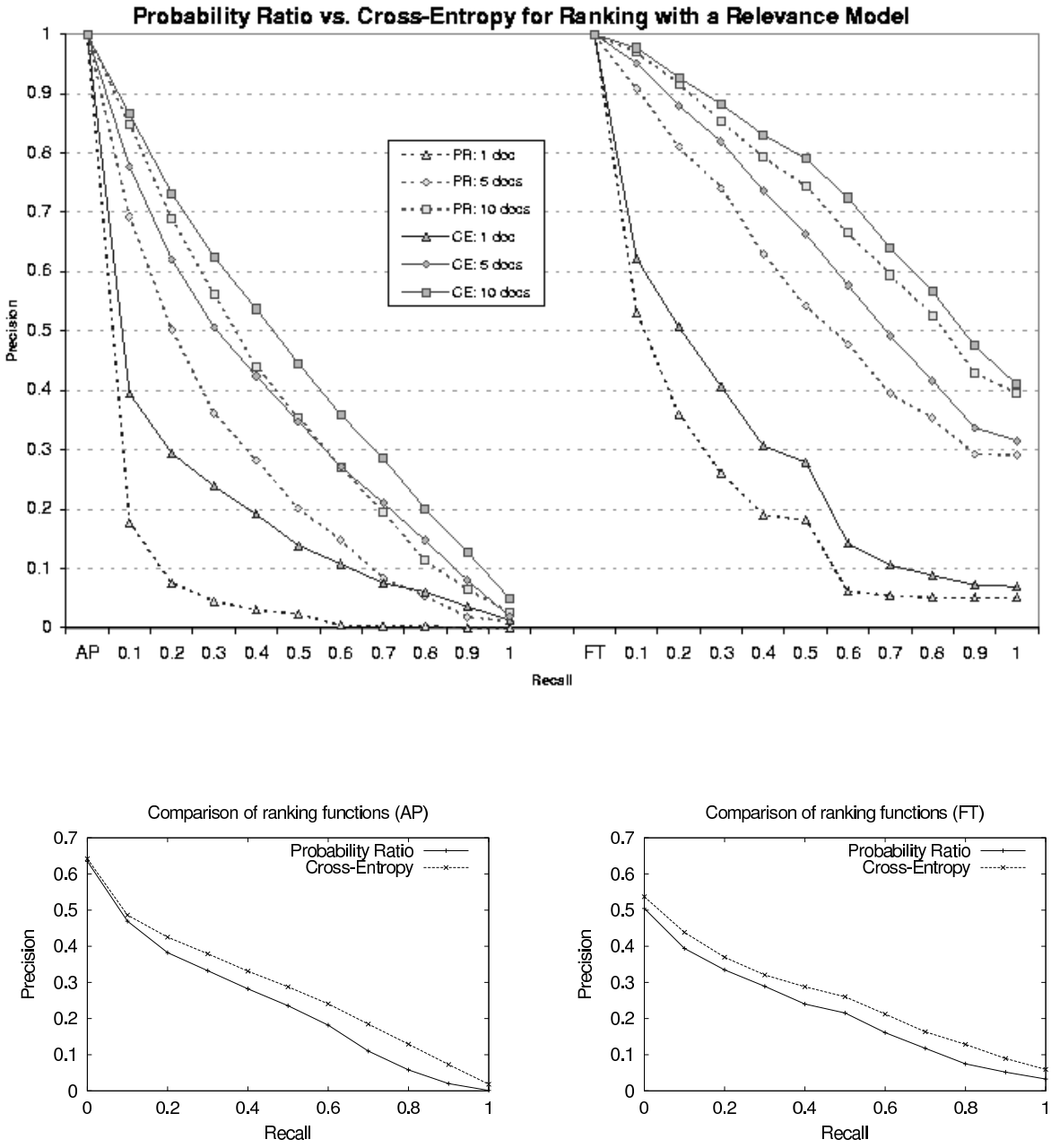


Figure 5.1. Comparing the effectiveness of two ranking functions: document likelihood (PR) and query likelihood (CE). We show results for two datasets: (AP) on the left side and (FT) on the right. Using query likelihood leads to noticeably higher precision at all levels of recall. The difference is consistent for relevance models of different quality: 1, 5 or 10 training documents (top) as well as relevance models estimated from the words in the query.

Corpus	Document likelihood	Query likelihood	% change
AP	0.228	0.272	+19%
FT	0.212	0.244	+15%
LA	0.226	0.254	+12%
WSJ	0.195	0.300	+54%
TDT	0.621	0.712	+15%

Table 5.5. Contrasting performance of document-likelihood and query-likelihood ranking. Query likelihood results in substantially better performance on all five datasets involved.

as expected, more training documents translates to better performance. Relevance models estimated from 5 documents exhibit higher precision than models constructed from 1 relevant document, using 10 documents leads to an even better performance. Second, rankings based on query likelihood (CE) noticeably outperform the document likelihood (PR). The precision of query likelihood rankings is higher at all levels of recall, regardless of the quality of the relevance model. The results are consistent across two datasets: Associated Press and Financial Times. Note that relevant documents used to estimate the models were not excluded from the ranking, so precision numbers are somewhat higher than would be expected.

We observe similar results if we compare the two ranking methods with relevance models estimated without any examples of relevant documents. In this case we start with a short query and use equation (5.6), as discussed in section 5.1.1.3. Once the model is computed we rank the documents $d \in \mathcal{C}$ using query-likelihood (cross-entropy) and the document likelihood (probability ratio). Results are presented in the lower half of Figure 5.1. As before, we observe that ranking by query likelihood results in noticeably higher precision at all levels of recall. The difference is consistent across the two datasets involved.

A more detailed empirical comparison of the two ranking methods is provided in Table 5.5. We report mean average precision on five different datasets. In every case query-likelihood leads to substantially superior performance. The improvement is particularly dramatic on the Wall Street Journal corpus with TREC queries 1-200. Improvements are statistically significant.

5.2 Relevance Feedback

Relevance feedback represents a natural extension of the ad-hoc searching scenario. In addition to the query $\mathbf{q}$, the user is going to provide us with a small set of documents he considers relevant to his information need. Our goal is to use the relevant documents to construct a better estimate of the relevance model $P_{\mathcal{R}}(\cdot)$, which will hopefully lead to more effective document ranking. Relevance feedback algorithms have a number of practical applications. Intuitions gained from relevance feedback experiments can serve a good purpose in the areas of text classification, information routing and filtering, and Topic Detection and Tracking.

5.2.1 Representation

As far as representation is concerned, relevance feedback is no different from ad-hoc retrieval. As before, the representation X will consist of string length, followed by $M-1$ words from the vocabulary. Document and query generating transforms will be defined exactly as in the ad-hoc case (equations 5.1 and 5.2). We will use the same modeling choices, making the string length uniformly distributed over $1 \ldots M-1$, and taking the parameter space Θ to be the space of all unigram distributions over the vocabulary. We will also use the same ranking criterion: relative entropy, as given by equation (5.8) respectively. The only difference from the ad-hoc case will come in the way we estimate the expected parameter vector Eu that represents the underlying relevance model $P_{\mathcal{R}}(\cdot)$.

Previously, the relevance model was estimated from a single example $\mathbf{q}$. Recall that we assumed that $\mathbf{q}$ was a random sample from $P_{\mathcal{R}}(\cdot)$, and constructed posterior expectation $E_{\mathbf{q}}u$ of the unigram parameters u conditioned on observing $\mathbf{q}$. Now, instead of a single random sample we have a set of k random samples, denote them $\mathbf{q}^1 \ldots \mathbf{q}^k$. We do not assume that these samples are homogeneous and talk about the same topic. In fact, each sample could potentially discuss a different aspect of the underlying information need. We will

in turn treat each $\mathbf{q}^i$ as a sample from the relevance model and construct the corresponding parameter expectation $E_{\mathbf{q}^i}u$. During ranking, the relevance model $P_{\mathcal{R}}$ will be represented by the *mean* expected parameter vector $E_{\mathbf{q}^{1\ldots k}}u$, defined as follows:

$$E_{\mathbf{q}^{1\ldots k}}u(v) = \frac{1}{k}\sum_{j=1}^{k} E_{\mathbf{q}^j}u(v) = \cdots = \frac{1}{k}\sum_{j=1}^{k}\frac{\sum_{\mathbf{d}} u_{\mathbf{d}}(v)\prod_{i=1}^{m_j} u_{\mathbf{d}}(q_{j,i})}{\sum_{\mathbf{d}}\prod_{i=1}^{m_j} u_{\mathbf{d}}(q_{j,i})} \tag{5.13}$$

Here $q_{j,i}$ denotes the i'th word in the j'th relevant example $\mathbf{q}^j$, and m_j is the total number of words in $\mathbf{q}^j$. Once we have the mean expectation from equation (5.13), we simply plug it into our favorite ranking criterion, be it document likelihood (equation 5.7) or model comparison (equation 5.8).

We would like to point out that equation (5.13) exhibits rather peculiar behavior when the relevant examples $\mathbf{q}^1 \ldots \mathbf{q}^k$ are included in the summation that goes over $\mathbf{d}$. Recall that $\sum_{\mathbf{d}}$ goes over all documents in our collection $\mathcal{C}$. If relevant examples are included in that summation, equation (5.13) becomes equivalent to the following very simple form:

$$\begin{aligned} E_{\mathbf{q}^{1\ldots k}}u(v) &= \frac{1}{k}\sum_{j=1}^{k}\sum_{\mathbf{d}} u_{\mathbf{d}}(v)\left(\frac{\prod_i u_{\mathbf{d}}(q_{j,i})}{\sum_{\mathbf{d}}\prod_i u_{\mathbf{d}}(q_{j,i})}\right) \\ &\approx \frac{1}{k}\sum_{j=1}^{k}\sum_{\mathbf{d}} u_{\mathbf{d}}(v)\cdot\delta(\mathbf{d},\mathbf{q}^j) \\ &= \frac{1}{k}\sum_{j=1}^{k} u_{\mathbf{q}^j}(v) \end{aligned} \tag{5.14}$$

The reason for this curious behavior is the fact that documents are *really long* strings of text. A typical document $\mathbf{q}^j$ in any TREC collection will easily contain several thousand words (including repetitions). This means that the product $\prod_i u_{\mathbf{d}}(q_{j,i})$ in equation (5.14) will involve multiplying thousands of rather small numbers, leading to infinitesimally small values. Of course, the denominator will also become infinitesimally small. The effect, aside from computational nuisance, is that the ratio inside the parenthesis becomes extremely peaked: it is nearly 1 when $\mathbf{d}$ is most similar to $\mathbf{q}^j$, and nearly zero for all other points $\mathbf{d}$. In other words, it starts to act as a delta function $\delta(\mathbf{d},\mathbf{q}^j)$. The result is that equation (5.14) simply picks out the relevant documents from the collection $\mathcal{C}$ and averages their empirical distributions. Note that the same does not happen in the ad-hoc scenario because: (i) queries are orders of magnitude shorter than relevant documents, and (ii) queries are rarely included as a kernel point $\mathbf{d}$ in the summation $\sum_{\mathbf{d}}$.

There are both positive and negative consequences stemming from equation (5.14). The upside is that we get a very simple formula for computing the expected relevant parameter vector. And for large values of k the formula is quite effective. The downside is that in some cases we really do not want the equation to behave that way. This will become particularly important later, when we dive into the Topic Detection and Tracking scenario.

5.2.2 Experiments

In this section we will briefly report experimental results for very simple applications of user feedback in the framework of relevance models. In the experiments that follow we assumed a very simplistic feedback setting. We assume that along with the query we get a set of 5 relevant documents, chosen chronologically from the beginning of the collection. We compare the following retrieval algorithms:

5.2.2.1 System description

1. **Language Model Baseline.** Use just the original query $Q = q_1 \ldots q_m$, perform retrieval using the standard language-modeling approach with smoothing based on the Dirichlet prior [154], and the smoothing parameter set to $\mu = 1000$.

2. **Language Model Feedback.** Use Ponte's formulation of relevance feedback for language models [101]. Apply equation (5.12) to the provided set of relevant documents. Select 5 words $w_1 \ldots w_5$ with the highest average log-likelihood ratio. Add these words to the original query to form the expanded query $Q^{exp} = q_1 \ldots q_m, w_1 \ldots w_5$. Use the expanded query to perform standard language model retrieval, using Jelinek-Mercer smoothing [154] with the smoothing parameter set to $\lambda = 0.9$.

3. **Relevance Model Feedback.** Estimate a relevance model as described in section (5.2.1). Rank the documents using the relative entropy between the relevance model and the empirical document model (equation 5.8). We used Jelinek-Mercer smoothing and the parameter λ was set to 0.1 for the document language model. Note that the original query was completely ignored.

5.2.2.2 Experimental results

Figure 5.2 shows the standard 11-point recall-precision curves for the above three algorithms. Experiments were performed on three datasets: Associated Press (AP), Financial Times (FT) and Los-Angeles Times (LA), refer to Table 5.1 for details. Five documents used for feedback were removed from evaluation in all cases.

We observe that on all three datasets both of the feedback algorithms noticeably outperform the strong language-modeling baseline. Precision is better at all levels of recall, and average precision is improved by 20-25%, which is considered standard for relevance feedback. Differences are statistically significant. If we compare the two feedback algorithms to each other, we see that their performance is almost identical. The differences are very small and not statistically significant. Note that the algorithm using relevance models completely ignored the original query, whereas Ponte's feedback used it as a basis for the expanded query. We believe the performance of the relevance modeling algorithm can be further improved by retaining the query and making use of the techniques we discussed in the ad-hoc scenario.

5.3 Cross-Language Retrieval

In a cross-language scenario, our documents are written in language A. The native language of the user is B, and consequently queries are issued in language B. For simplicity, let's assume A stands for Arabic and B stands for Bulgarian. Our goal is to start with a Bulgarian query and return relevant Arabic documents. This may seem like an odd scenario, but it does have a number of useful applications, and has attracted a significant amount of research over the years. To give some justification for this scenario, we envision that the user has access to a human translator or to a machine translation engine, which can translate a limited amount of text from Arabic into Bulgarian. However, the cost (monetary or computational) of translation is too high to translate the entire collection into into Bulgarian. A natural solution is to perform cross-language retrieval and then translate only the retrieved documents.

5.3.1 Representation

As in the ad-hoc case, we assume that neither documents nor queries have any structure to them: documents are natural-language strings over the Arabic vocabulary $\mathcal{V}_A$, queries are short strings over the Bulgarian vocabulary $\mathcal{V}_B$. We assume that in its latent, unobserved form, each Arabic document comes with a ready translation into Bulgarian. So, in effect, each document is represented as two strings: one in Arabic, another in Bulgarian. The same is true for queries: each query originates as a long narrative in Bulgarian, along with its translation to Arabic. Let M_A be the upper bound on the length of Arabic strings, let M_B be a similar upper bound for Bulgarian. A latent representation of documents and queries will take the following common form: $X = n_A, a_1 \ldots a_{M_A}, n_B, b_1 \ldots b_{M_B}$. Here n_A and n_B represent the target length of description in Arabic and Bulgarian languages respectively. a_i represent all the words in the Arabic description, and b_j stand for the words from the corresponding description in Bulgarian. The entire representation space is: $\mathcal{S} = \{1 \ldots M_A\} \times \mathcal{V}_A^{M_A} \times \{1 \ldots M_B\} \times \mathcal{V}_B^{M_B}$.

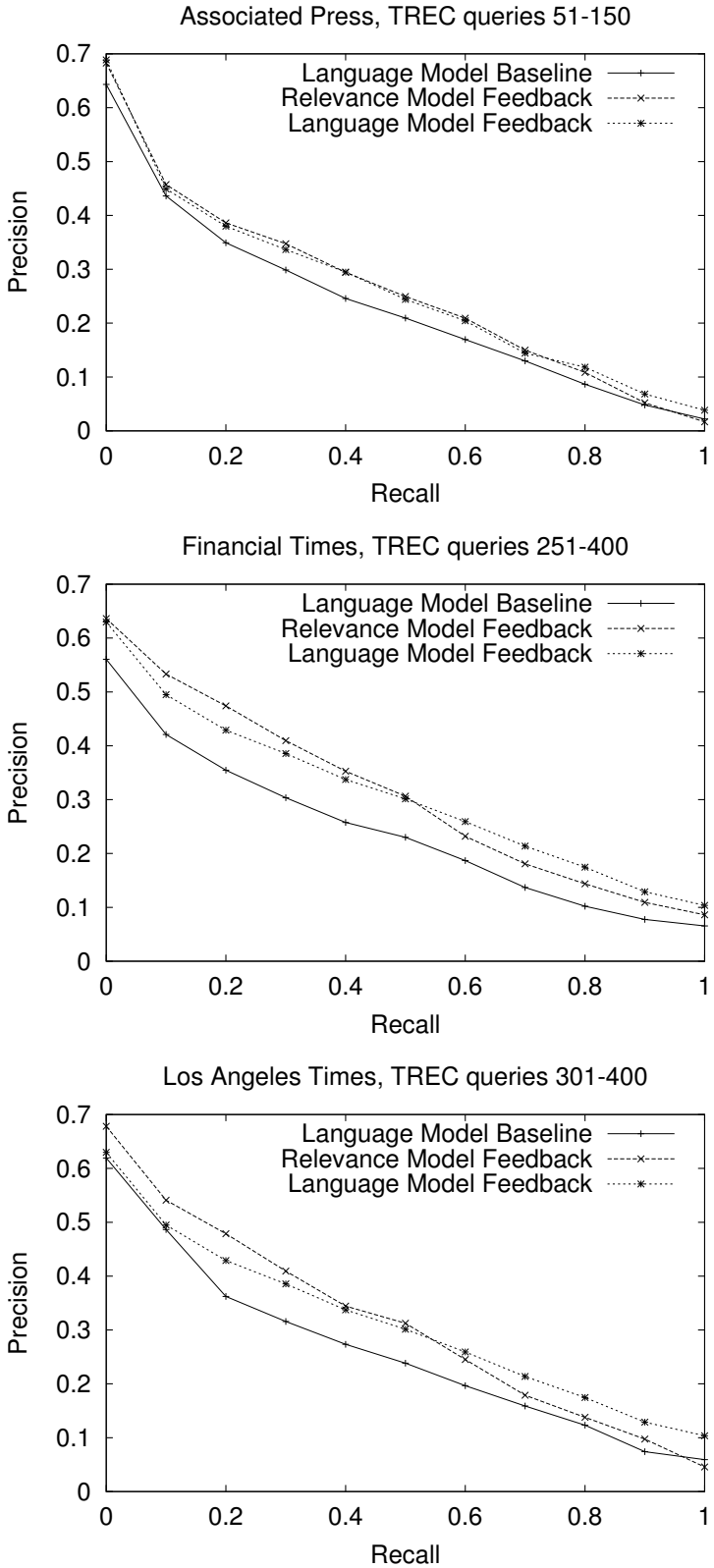


Figure 5.2. Relevance Feedback: relevance model performs as well as the feedback mechanism proposed by Ponte. Both feedback methods significantly outperform a strong baseline.

5.3.1.1 Document and query representation

The document and query generating transforms will then be defined as follows:

$$D(X) = D(n_A, a_1 \ldots a_{M_A}, n_B, b_1 \ldots b_{M_B}) = n_A, a_1 \ldots a_{n_A} \quad (5.15)$$

$$Q(X) = Q(n_A, a_1 \ldots a_{M_A}, n_B, b_1 \ldots b_{M_B}) = m, b_1 \ldots b_m \quad (5.16)$$

The document transform D simply returns the Arabic portion of the latent representation, truncated to the appropriate length. The query transform Q discards the Arabic portion entirely and selects m keywords from the Bulgarian portion. Keyword selection is performed by the same heuristic that was employed in the ad-hoc retrieval scenario (section 5.1.1).

5.3.1.2 Components of the relevance model

We assume that length of the Arabic description is uniformly distributed over $\{1 \ldots M_A\}$, and similarly length of the Bulgarian description is uniform over $\{1 \ldots M_B\}$. As in the ad-hoc scenario, we will use multinomial distributions as a natural model for words in the Arabic and Bulgarian parts of the representation. What makes the cross-language case different is the fact that we have two different vocabularies: $\mathcal{V}_A$ and $\mathcal{V}_B$. These vocabularies cannot be collapsed into one, since we want to keep the Bulgarian and Arabic strings separate. As a result, we will have to define two separate multinomial distributions for each point in the parameter space. θ will be a concatenation of two vectors: $\theta = \{\alpha, \beta\}$. The first vector will have dimensions $\alpha(a)$, defining the probabilities for every Arabic word a. The second vector will have dimensions $\beta(b)$, one for each Bulgarian word b. The space of all possible parameter settings will be the cross-product: $\Theta = I\!P_{\mathcal{V}_A} \times I\!P_{\mathcal{V}_B}$, where $I\!P_{\mathcal{V}_A} = \left\{\vec{x} \in [0,1]^{\mathcal{V}_A} : \sum_v x_v = 1\right\}$ is the simplex of all possible distributions over the Arabic words, and $I\!P_{\mathcal{V}_B}$ is the similar simplex for the Bulgarian vocabulary. As in the ad-hoc case, we will use a kernel-based density p_{ker} over the parameter space Θ, and will again opt for delta kernels (equation 4.29). Under the above definitions we will get the following expression for the probability of observing some information item X from the relevance model:

$$\begin{aligned} P_{\mathcal{R}}(n_A, a_{1\ldots M_A}, n_B, b_{1\ldots M_B}) &= \frac{1}{M_A}\frac{1}{M_B} \int_{I\!P_{\mathcal{V}_A} \times I\!P_{\mathcal{V}_B}} \left(\prod_{i=1}^{n_A} \alpha(a_i)\right) \left(\prod_{i=1}^{n_B} \beta(b_i)\right) p_{ker,\delta}(d\alpha \times d\beta) \\ &= \frac{1}{M_A}\frac{1}{M_B}\frac{1}{N} \sum_{\mathbf{w}} \left(\prod_{i=1}^{n_A} \alpha_{\mathbf{w}}(a_i)\right) \left(\prod_{i=1}^{n_B} \beta_{\mathbf{w}}(b_i)\right) \end{aligned} \quad (5.17)$$

The summation goes over every training example $\mathbf{w}$. N is the total number of training strings. The exact construction of the training set will be discussed below. To get the probability of observing a a document $D{=}d$ we simply restrict equation (5.17) to the Arabic part of the space by removing M_B and the product over the Bulgarian words b_i. To get the probability of a Bulgarian query, we restrict the product to just the query terms. Both of these steps are justified by our arguments in section 3.2.2.2.

5.3.1.3 Parameter estimation with a parallel corpus

The core part of the relevance model given by equation (5.17) is a set of distributions $\alpha_{\mathbf{w}}(\cdot)$ and $\beta_{\mathbf{w}}(\cdot)$. These are supposed to be empirical distributions corresponding to some collection of training examples. $\alpha_{\mathbf{w}}(\cdot)$ is a distribution over the Arabic vocabulary, $\beta_{\mathbf{w}}(\cdot)$ is the corresponding distribution over Bulgarian words. Note that these distributions are tied to each other: they reflect word usage in the same training example $\mathbf{w}$. Such training examples are certainly consistent with our representation space $\mathcal{S}$: recall that we assumed that the latent form of each document contains both Arabic and Bulgarian narrative. But in reality the documents in our collection $\mathcal{C}$ are written in Arabic, the Bulgarian translation simply will not be present. Consequently, we cannot use our collection $\mathcal{C}$ as a set of training examples. What we need is a collection that does include both Arabic and Bulgarian renditions of the same conceptual narrative. Such collections are called *parallel* or *comparable* corpora, and are the staple of research in statistical machine translation [17]. There is a slight difference between parallel and comparable corpora. The former would contain an *exact* translation of each Arabic sentence into its Bulgarian equivalent. The latter does not require that sentences be exact translations, as long as groups of sentences (documents) in Arabic carry roughly the same semantic content

as groups of sentences in Bulgarian. Comparable corpora are somewhat easier to obtain, but they cannot be used for training machine translation systems. Comparable corpora are sufficient for our purposes, since we are interested in topical correlations across the languages, and not in surface translation.

For the purposes of computing equation (5.17), we assume that we have access to a comparable corpus $\mathcal{C}_{train}$. This corpus consists of paired strings $\mathbf{w} = \{\mathbf{a}, \mathbf{b}\}$, where $\mathbf{a}$ is some Arabic document and $\mathbf{b}$ is a corresponding Bulgarian rendition. Empirical distributions $\alpha_{\mathbf{w}}$ and $\beta_{\mathbf{w}}$ corresponding to a specific pair $\mathbf{w}$ would be defined as:

$$\begin{aligned} \alpha_{\mathbf{w}}(a) &= \lambda \frac{n(a,\mathbf{a})}{|\mathbf{a}|} + (1-\lambda) \frac{\sum_{\mathbf{a}} n(a,\mathbf{a})}{\sum_{\mathbf{a}} |\mathbf{a}|} \\ \beta_{\mathbf{w}}(b) &= \lambda \frac{n(b,\mathbf{b})}{|\mathbf{b}|} + (1-\lambda) \frac{\sum_{\mathbf{b}} n(b,\mathbf{b})}{\sum_{\mathbf{b}} |\mathbf{b}|} \end{aligned} \tag{5.18}$$

Here $n(a, \mathbf{a})$ is the number of times the word a occurs in Arabic document $\mathbf{a}$ and $|\mathbf{a}|$ is the length of that document. The summation involving $\mathbf{a}$ goes over all Arabic training strings. Bulgarian definitions are identical.

5.3.1.4 Parameter estimation with a dictionary

Parallel and comparable corpora are often difficult to come by. Translation is a rather expensive process, and we may not be able to find a parallel corpus for the pair of languages that we are dealing with. For example, we are not aware of the existence of any Arabic / Bulgarian parallel corpora. How can we estimate the paired unigrams $\{\alpha_{\mathbf{w}}, \beta_{\mathbf{w}}\}$ in this case?

If a parallel corpus is not available, we may have access to a statistical dictionary. A statistical dictionary is a matrix $T(a, b)$, which gives a likelihood that Arabic word a would be translated into Bulgarian word b. Any existing dictionary can be turned into a statistical dictionary by uniformly distributing the translation mass over all Bulgarian words b listed as translations of a. In this case, given an Arabic distribution $\alpha_{\mathbf{w}}(\cdot)$ we can define the corresponding Bulgarian distribution as follows:

$$\beta_{\mathbf{w}}(b) = \sum_{a \in \mathcal{V}_A} \alpha_{\mathbf{w}}(a) \cdot T(a,b) \tag{5.19}$$

Equation (5.19) forms the basis of the *translation* approach to ad-hoc [11] and cross-language retrieval [149, 150]. In this case, we will take the available collection of Arabic documents $\mathcal{C}$ as our training collection $\mathcal{C}_{train}$. For each Arabic document $\mathbf{w}$ we will define the Arabic distribution $\alpha_{\mathbf{w}}(\cdot)$ according to equation (5.18), and the corresponding Bulgarian distribution $\beta_{\mathbf{w}}(\cdot)$ according to equation (5.19). The resulting collection of paired distributions $\{\alpha_{\mathbf{w}}, \beta_{\mathbf{w}}\}$ will be used in equation (5.17).

5.3.1.5 Document ranking

We will use model comparison (section 3.2.3.3) for ranking documents in response to the user's query. Recall that model comparison involves computing relative entropy between the expected relevance model and the empirical distribution of a document. In our case, a document gives an empirical distribution over Arabic words, so we will compute relative entropy over the Arabic vocabulary and rank the documents according to:

$$-D(E_q\theta || \theta_d) \propto \sum_{a \in \mathcal{V}_A} E_q\theta(a) \log \theta_d(a) \tag{5.20}$$

The empirical distribution $\theta_d(\cdot)$ will be computed according to equation (5.18), the expected relevance model $E_q\theta$ will take the following form:

$$\begin{aligned} E_q\theta(a) &= \frac{\int_{I\!P_{\mathcal{V}_A}} \int_{I\!P_{\mathcal{V}_B}} \alpha(v) \left\{\prod_{i=1}^m \beta(q_i)\right\} p_{ker,\delta}(d\alpha \times d\beta)}{\int_{I\!P_{\mathcal{V}_A}} \int_{I\!P_{\mathcal{V}_B}} \left\{\prod_{i=1}^m \beta(q_i)\right\} p_{ker,\delta}(d\alpha \times d\beta)} \\ &= \frac{\sum_{\mathbf{w}} \alpha_{\mathbf{w}}(a) \prod_{i=1}^m \beta_{\mathbf{w}}(q_i)}{\sum_{\mathbf{w}} \prod_{i=1}^m \beta_{\mathbf{w}}(q_i)} \end{aligned} \tag{5.21}$$

The summation $\sum_{\mathbf{w}}$ goes over all paired string $\mathbf{w} = \{\mathbf{a}, \mathbf{b}\}$ in our training collection. Recall that the training collection can correspond either to a parallel corpus, or to the Arabic collection $\mathcal{C}$ augmented with a statistical dictionary. Equation (5.21) works equally well for both cases.

	LDC	CETA	HK News	Combined
English terms	86,000	35,000	21,000	104,997
Chinese terms	137,000	202,000	75,000	305,103

Table 5.6. Composition of the BBN bilingual lexicon

	HK News	TDT	HK News + TDT
Document pairs	18,147	46,692	64,839
English terms	28,806	67,542	83,152
Chinese terms	49,218	111,547	132,453

Table 5.7. Composition of the parallel corpus used in our experiments.

5.3.2 Experiments

The problem of cross-language information retrieval mirrors the problem of adhoc retrieval with one important distinction: the query $Q = q_1 \ldots q_k$ is given in a language that is different from the collection of documents $\mathcal{C}$. In the previous section we discussed how we can use a parallel corpus or a statistical dictionary to estimate the relevance model $E_{\mathbf{q}}\theta$ in the target language. Once a relevance model is computed, we can rank the documents using equation (5.20). In this section we carry out an evaluation of this approach on the cross-language (English - Chinese) retrieval task of TREC9.

5.3.2.1 Chinese resources

All of our cross-language experiments were performed on the dataset used in the TREC9 cross-lingual evaluation. The dataset consists of 127,938 Chinese documents, totaling around 100 million characters. We used the official set of 25 queries. We used two query representations: *short* queries used only the title field, while *long* queries used title, description and narrative fields.

Experiments involving a bilingual dictionary used the statistical lexicon created by Xu et.al [150]. The lexicon was assembled from three parts: the LDC dictionary, the CETA dictionary, and the statistical dictionary, learned from the Hong-Kong News corpus by applying the GIZA machine translation toolkit. Table 5.6 provides a summary of the dictionary components.

In the experiments that made use of the parallel corpus, we used the Hong-Kong News parallel dataset, which contains 18,147 news stories in English and Chinese. Because it is so small, the Hong-Kong parallel corpus has a significant word coverage problem. In order to alleviate the problem, we augmented the corpus with the TDT2 and TDT3 [24] pseudo-parallel datasets. These corpora contain 46,692 Chinese news stories along with their SYSTRAN translations into English. Since the documents are translated by software, we do not expect the quality of the TDT corpus to be as high as Hong-Kong News. We discuss the impact of adding the TDT corpus in section 5.3.2.6. The composition of the parallel corpus is detailed in Table 5.7.

5.3.2.2 Chinese pre-processing

The pre-processing performed on the Chinese part of the corpus was very crude, due to our limited knowledge of the language. The entire dataset, along with the Chinese queries was converted into the simplified encoding (GB). We carried out separate experiments with three forms of tokenization: (i) single Chinese characters (unigrams), (ii) half-overlapping adjacent pairs of Chinese characters (bigrams), and (iii) Chinese "words", obtained by running a simple dictionary-based segmenter, developed by F. F. Feng at the University of Massachusetts. In the following sections we will report separate figures for all three forms of tokenization, as well as a linear combination of them. We did not remove any stop-words, or any punctuation characters from either Chinese documents or queries. This results in some spurious matches and also in these characters figuring prominently in the relevance models we constructed.

5.3.2.3 System descriptions

We compare retrieval performance of the following models:

LM **Mono-lingual baseline.** We use the basic language modeling system, which was reported as a baseline in a number of recent publications [150, 143]. The system is identical to the **LM** system described in section 5.1.2.3. Chinese documents D are ranked according to the probability that a Chinese query $c_1 \ldots c_k$ was generated from the language model of document D.

RM **Mono-lingual Relevance Model.** This system is included as an alternative mono-lingual baseline, and to demonstrate the degree to which relevance models degrade, when estimated in a cross-lingual setting. The system is identical to the **RM** system from section 5.1.2.3. Given a Chinese query $c_1 \ldots c_k$, we compute the relevance model as in the ad-hoc scenario. We use relative entropy as the ranking function.

TM **Probabilistic Translation Model.** As a cross-lingual baseline, we report the performance of our implementation of the system used by Xu et.al. [150]. The translation model was originally proposed by Berger and Lafferty [11] and Hiemstra and de Jong [57]. We used the formulation advocated by Xu et al. [150]. We used the same statistical lexicon and the same system parameters that were reported in [150].

pRM **Cross-lingual Relevance Model (parallel).** Given an English query $e_1 \ldots e_k$, we estimate a relevance model in Chinese using techniques suggested in section 5.3.1.3. For probability estimation we use the combined parallel corpus (see Table 5.7). The Chinese documents are then ranked by their relative entropy with respect to the relevance model.

dRM **Cross-lingual Relevance Model (dictionary).** We estimate the cross-lingual relevance model as suggested in section 5.3.1.4 Equation (5.19) is used to compute the probability of an English word e_i given a Chinese document $\mathbf{d}$. We use the lexicon reported in [150] for translation probabilities. Relative entropy is used as the ranking method.

In all cases we performed separate experiments on the three representations of Chinese: unigrams, bigrams and "words". Refer to section 5.3.2.2 for details. The smoothing parameters λ were tuned separately for each representation as we found that smoothing affects unigrams and bigrams very differently. The results from the three representations were then linearly combined. The weights attached to each representation were set separately for every model, in order to show best results. As an exception, the Probabilistic Translation Model was evaluated on the same representation that was used by Xu et.al.[150]. Due to the absence of the training corpus, the tuning of all parameters was performed on the testing data using a brute-force hill-climbing approach. The small number of queries in the testing dataset precluded the use of any statistical significance tests.

5.3.2.4 Baseline results

Table 5.8 shows the retrieval performance, of the described models on the TREC9 cross-language retrieval task. We use non-interpolated average precision as a performance measure. Percentage numbers indicate the difference from the mono-lingual baseline. We show results for both short and long versions of the queries. Our monolingual results form a strong baseline, competitive with the results reported by [143, 150]. This is somewhat surprising, since our processing of Chinese queries was very simplistic, and a lot of spurious matches were caused by punctuation and stop-words in the queries. We attribute the strong performance to the careful selection of smoothing parameters and combination of multiple representations. The monolingual relevance model provides an even higher baseline for both short and long queries.

The Probabilistic Translation Model achieves around 85% - 90% percent of the mono-lingual baseline. Xu et al. in [150] report the performance of the same model to be somewhat higher than our implementation (0.3100 for long queries). We attribute the differences to the different form of pre-processing used by Xu et al, since we used the same bilingual lexicon and the same model parameters as [150].

5.3.2.5 Cross-lingual relevance model results

Table 5.8 shows that cross-lingual relevance models perform very well, achieving 93% - 98% of the monolingual baseline on the combined representation. This performance is better than most previously-reported results [143, 150], which is somewhat surprising, given our poor pre-processing of Chinese. Our model

	Unigrams		Bigrams		Words		Combination	
Short Queries								
LM (mono)	0.244		0.237		0.260		0.287	
RM (mono)	0.240	-2%	0.252	+6%	0.307	+18%	0.310	+8%
TM	—		—		—		0.254	-11%
dRM (dictionary)	0.277	+13%	0.268	+13%	0.277	+7%	0.280	-2%
pRM (parallel)	0.225	-8%	0.251	+6%	0.249	-4%	0.267	-7%
Long Queries								
LM (mono)	0.275		0.309		0.283		0.330	
RM (mono)	0.283	+3%	0.341	+10%	0.324	+14%	0.376	+14%
TM	—		—		—		0.276	-16%
dRM (dictionary)	0.300	+9%	0.307	-1%	0.317	+12%	0.318	-4%

Table 5.8. Average Precision on the TREC9 cross-language retrieval task. Cross-lingual relevance models perform around 95% of the strong mono-lingual baseline

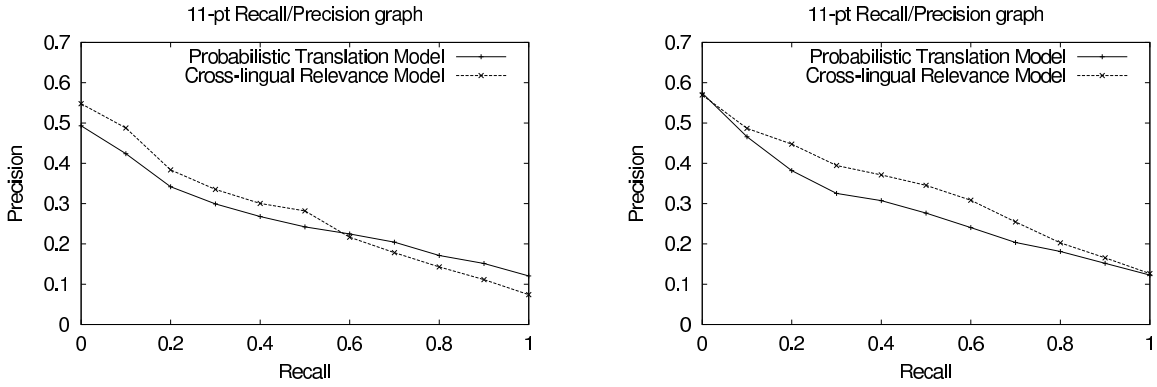


Figure 5.3. Cross-lingual relevance models outperform the Probabilistic Translation Model on both the short (left) and long (right) queries.

noticeably outperforms the Probabilistic Translation Model on both long and short queries (see figure 5.3). It is also encouraging to see that cross-lingual relevance models perform very well on different representations of Chinese, even though they do not gain as much from the combination as the baselines.

5.3.2.6 Importance of coverage

Note that relevance models estimated using a bilingual lexicon perform better than the models estimated from the parallel corpus. We believe this is due to the fact that our parallel corpus has an acute coverage problem. The bilingual dictionary we used covers a significantly larger number of both English and Chinese words. In addition, two thirds of our parallel corpus was obtained using automatic machine translation software, which uses a limited vocabulary. It is also worth noting that the remaining part of our parallel corpus, Hong-Kong News, was also used by Xu et al. [150] in the construction of their bilingual dictionary. Refer to Table 5.7 for details.

Table 5.10 illustrates just how serious the coverage problem is. We show performance of Relevance Models estimated using just the Hong-Kong News portion of the corpus, versus performance with the full corpus. We observe tremendous improvements of over 100% which were achieved by adding the TDT data, even though this data was automatically generated using SYSTRAN.

Prec.	LM	RM		TM		dRM		pRM	
5 docs	0.288	0.336	+17%	0.256	-11%	0.320	+11%	0.352	+22%
10 docs	0.260	0.288	+11%	0.212	-19%	0.232	-11%	0.288	+11%
15 docs	0.224	0.242	+8%	0.186	-17%	0.208	-7%	0.245	+10%
20 docs	0.212	0.222	+5%	0.170	-20%	0.196	-8%	0.208	-2%
30 docs	0.186	0.190	+2%	0.144	-23%	0.169	-9%	0.182	-2%

Table 5.9. Initial precision on the TREC9 CLIR task. Cross-lingual Relevance Models noticeably outperform the mono-lingual baselines.

Average Precision	HK News	HK News + TDT	
Unigrams	0.1070	0.2258	+111%
Bigrams	0.1130	0.2519	+123%
"Words"	0.1210	0.2493	+106%

Table 5.10. Parallel corpus size has a very significant effect on the quality of Cross-lingual Relevance Models. Adding the pseudo-parallel TDT corpus more than doubled the average precision.

5.3.2.7 High-precision performance

Average precision is one of the most frequently reported metrics in cross-language retrieval. This metric is excellent for research purposes, but it is also important to consider user-oriented metrics. Table 5.9 shows precision at different ranks in the ranked list of documents. Precision at 5 or 10 documents is what affects a typical user in the web-search setting. We observe that Cross-lingual Relevance Models exhibit exceptionally good performance in this high-precision area. Models estimated using the parallel corpus are particularly impressive, outperforming the mono-lingual baseline by 20% at 5 retrieved documents. Models estimated from the bilingual dictionary perform somewhat worse, though still outperforming mono-lingual performance at 5 documents. Both estimation methods outperform the Probabilistic Translation Model. We consider these results to be extremely encouraging, since they suggest that Cross-lingual Relevance Models perform very well in the important high-precision area.

5.3.3 Significance of the cross-language scenario

The cross-language retrieval scenario discussed in the present section serves a very important purpose in the development of our model. We have demonstrated that our model can discover dependencies between words in two different languages. The generative approach allows us to associate a given bag of words (an English query) with a different bag of words (a Chinese document). The two sets of words can come from completely different vocabularies, as long as we have a parallel corpus to help us learn the coupling between them. This capacity opens a fascinating array of possible applications for our model. Our method should be able to associate any two sets of variables, as long as two conditions hold: **(i)** we have appropriate training data and **(ii)** variables in question look like words in a language.

In the following sections we will discuss three multimedia applications which extend the cross-language scenario. In each case our goal will be to associate English words with some non-linguistic features of the data. We will transform these features into something that looks like words and then use exactly the same techniques that proved to be so successful in the current section.

5.4 Handwriting Retrieval

Our first multimedia application concerns retrieval of manuscripts. A surprisingly large number of historical documents is available in handwritten form. Some of these documents are political writings, like the letters of George Washington and Thomas Jefferson, other documents represent scientific works, such as the works of Isaac Newton. These manuscripts are often available in an electronic form, but in a form of an image rather than searchable text. Finding a certain item a collection of manuscripts is a tedious task because these collections are often quite large, and transcripts are not always available for every document. Even when a

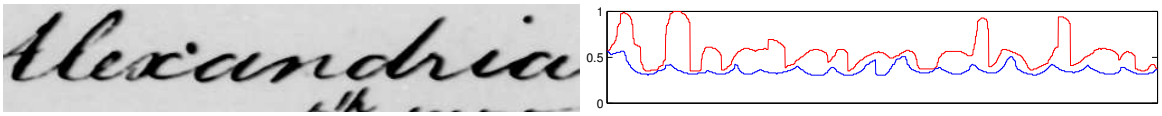


Figure 5.4. Left: a sample handwritten word from the collection of George Washington's manuscripts. Right: de-slanted upper and lower profiles of the word on the left. The profiles serve as a basis for the discrete features in our algorithm.

transcript is available, it is often of little use because there may not be a clear alignment between the transcript and the scanned images. The alignment is particularly important to historians who prefer to work with the original pages of the manuscript, rather than with the transcript. For example, if our historian was looking for all letters written by George Washington on a certain topic, he or she would not have an easy way of fetching images that contain the words she is interested in finding. This scenario gives rise to the problem of handwriting retrieval, which will be addressed in the present section.

Before we proceed any further we must clear away a common misconception. Handwriting retrieval cannot be trivially solved by applying Optical Character Recognition (OCR). The level of noise in historical manuscripts is too high for successful application of OCR. For example, if we apply an off-the-shelf OCR system to a manuscript, we would expect to get complete garbage as output. Commercial OCR depends on being able to separate words into distinct characters, which is not possible with handwriting. If instead of OCR we use prototype handwriting recognition systems, we would expect a word error rate around 60-70%.[1] This means that two out of three words in the transcript would be wrong. There are a number of reasons for such poor performance. In some cases the original manuscripts are highly degraded. Often pages are scanned from microfilm rather than the original manuscript; re-scanning is not an option for fear of further degradation of a valuable historic article. Finally, the fluid cursive of historical manuscripts is quite different from the present day handwriting which contains many printed characters.

5.4.1 Definition

A manuscript archive is a collection of images, each representing a complete page of handwriting. Our goal is to be able to retrieve sections of these pages in response to a text query. This task is quite different from a transcription task.

The first step in dealing with a scanned page is to identify the regions containing individual words. This is not a trivial task, but there exist a number of successful techniques, for details see [81, 103, 76]. We are going to assume that the pages have already been segmented, so we are faced with a collection of small images, each image containing a single handwritten word. An example of such an image is shown in the left portion of Figure 5.4. Naturally, each word image corresponds to a certain word from an English vocabulary, the example in Figure 5.4 is the word *"Alexandria"*. The connection between the image and the word is trivial for a human observer, but, as we argued previously, poses a significant algorithmic challenge. In the following sections we will discuss a system that is capable of retrieving a bitmap image shown in Figure 5.4 in response to the textual query *"Alexandria"*.

5.4.2 Representation

We have a collection $\mathcal{C}$ of pre-segmented word images. Using the terminology from previous scenarios, each bitmap is a document. A query, in the simplest case, is a single English word. The goal is to retrieve documents (bitmaps) that would be recognized by the user as handwritten renditions of the query.

We assume that the latent, un-observed representation X of each document contains a bitmap image $\mathbf{d}$ along with the corresponding word w from the English vocabulary $\mathcal{V}$. The query generating transform Q is trivial: it takes the pair $\{\mathbf{d}, e\}$ and returns the word e. The document transform D is considerably more complex. We will take the bitmap $\mathbf{d}$ and convert it to a set of discrete *features* $f_1 \ldots f_k$. These features are intended to convey useful information about the overall shape of the word; description of the features will

[1] Higher accuracy can be achieved in live handwriting recognition, where the system has access to pen stroke, position and velocity.

be provided below. The important thing to note is that the features f_i are discrete, i.e. they take values in some finite set $\mathcal{F}$, which can be thought of as the feature *vocabulary*. This vocabulary contains items that presumably bear some semantic content, similar to the words in a normal language. If we continue this line of thinking, we will see that the feature representation $\{f_1 \ldots f_k\}$ is essentially an *utterance* in some strange language $\mathcal{F}$. This is an important observation because it allows us to relate the problem of handwriting retrieval to the problem of cross-language retrieval, which we successfully addressed in the previous section. Finding associations between the English word e and feature words $f_1 \ldots f_k$ is no different than finding associations between English queries and words in Chinese documents.

5.4.2.1 Features and discretization

The word shape features we use in this work are described in [76]. The features include: **(i)** height, width, area and aspect ratio of the bounding box containing the word, **(ii)** an estimate for the number of *ascenders* (letters like 'd', 'b' or 'l') and *descenders* ('p', 'q' or 'y') in the word, and **(iii)** Fourier coefficients from the DFT of the upper and lower word shape profiles shown in the right portion of Figure 5.4. This feature set allows us to represent each image of a handwritten word with a continuous-space feature vector of constant length.

With these feature sets we get a 26-dimensional vector for word shapes. These representations are in continuous-space, but the model requires us to represent all feature vectors in terms of a discrete vocabulary of fixed size. One possible approach to discretization [63] is to cluster the feature vectors. Each cluster would correspond to one term in the feature vocabulary $\mathcal{F}$. However, this approach is rather aggressive, since it considers words or shapes to be equal if they fall into the same cluster.

We chose a discretization method that preserves a greater level of detail, by separately binning each dimension of a feature vector. Whenever a feature value falls into a particular bin, an associated bin number is added to the discrete-space representation of the word or shape. We used two overlapping binning schemes - the first divides each feature dimension into 10 bins while the second creates an additional 9 bins shifted by half a bin size. The overlapping bins are intended to avoid the undesirable boundary effects which happen when two very close real numbers fall on two different sides of a binning boundary. After discretization, we end up with 52 discrete features word image. The entire vocabulary $\mathcal{F}$ contains $26 \cdot 19 = 494$ entries.

5.4.2.2 Components of the relevance model

We can define the relevance distribution $P_{\mathcal{R}}$ in exactly the same way as for the cross-language retrieval scenario. The only difference is that now all sequences involved have exactly the same length: we always have 1 English word and 52 feature words. Both components will be modeled by the multinomial distributions over their respective vocabulary. Accordingly, the parameter space Θ is given by the cross product $I\!P_{\mathcal{V}} \times I\!P_{\mathcal{F}}$, where $I\!P_{\mathcal{V}}$ is the set of all distributions over the English vocabulary $\mathcal{V}$, and $I\!P_{\mathcal{F}}$ is the set of all distributions over the feature vocabulary $\mathcal{F}$. As before, we will adopt a kernel-based density p_{ker} with delta kernels (equation 4.29). The probability of a complete observation X (an image together with the transcribed word) is:

$$\begin{aligned} P_{\mathcal{R}}(e, f_1 \ldots f_k) &= \int_{I\!P_{\mathcal{V}}} \int_{I\!P_{\mathcal{F}}} \alpha(e) \left(\prod_{i=1}^{k} \beta(f_i) \right) p_{ker,\delta}(d\alpha \times d\beta) \\ &= \frac{1}{N} \sum_{\mathbf{x}} \alpha_{\mathbf{x}}(e) \left(\prod_{i=1}^{k} \beta_{\mathbf{x}}(f_i) \right) \end{aligned} \tag{5.22}$$

The summation goes over all examples $\mathbf{x}$ in the training set. $\alpha_{\mathbf{x}}(\cdot)$ and $\beta_{\mathbf{x}}(\cdot)$ represent the distributions over the English words ($\mathcal{V}$) and the feature words ($\mathcal{F}$) respectively. We compute the estimates for these distributions using equation (5.18) – the one we used in the cross-language scenario.

5.4.2.3 Document ranking

We will describe two retrieval scenarios. In the first, we are only interested in retrieving single-word images, it is useful when a historian wants to pinpoint all locations of a given word, such as *"Alexandria"*. Recall

that observable documents $\mathbf{d}$ in our collection contain only the features $f_1 \ldots f_k$, they are not annotated with the corresponding word e. However, we can use equation (5.22) to come up with the probability that e would be used as a transcription, conditioned on observing the word shape features $f_1 \ldots f_k$:

$$E_{\mathbf{d}}\theta(e) \quad = \quad \frac{\sum_{\mathbf{w}} \alpha_{\mathbf{x}}(e) \prod_{i=1}^{k} \beta_{\mathbf{x}}(f_i)}{\sum_{\mathbf{x}} \prod_{i=1}^{k} \beta_{\mathbf{x}}(f_i)} \tag{5.23}$$

Equation (5.23) allows us to perform two important tasks:

- we can use it to *annotate* a word image $\mathbf{d}$ with likely English transcription by picking one or more words e that yield a high value of $E_{\mathbf{d}}\theta(e)$.
- we can use it to *rank* different word images in response to a single-word query e; images $\mathbf{d}$ should be ranked in the order of decreasing $E_{\mathbf{d}}\theta(e)$.

The above scenario does not allow us to use multi-word queries like *"Fort Cumberland"*, which may also be quite interesting in historical analysis. To get around this inconvenience we will describe a way to rank entire passages of handwritten text. Let $\mathbf{d}^1 \ldots \mathbf{d}^n$ be a sequence of word images, representing a line or perhaps a paragraph from the manuscript. We can construct a language model for the entire sequence by averaging the individual distributions $E_{\mathbf{d}^i}\theta(e)$ that we get from equation (5.23). After that we can use query likelihood criterion to rank entire lines in response to a multi-word query $e_1 \ldots e_m$:

$$P(e_1 \ldots e_m) = \prod_{j=1}^{m} \sum_{i=1}^{n} E_{\mathbf{d}^i}\theta(e_j) \tag{5.24}$$

Our use of query likelihood was prompted by reasons of computational efficiency – with proper algorithmic optimization it is by far the fastest of all ranking criteria.

5.4.3 Experiments

Experiments discussed in this section have been previously reported in [103]. We will discuss two types of evaluation. First, we briefly look at the annotation capability of the annotation as outlined above. We train a model on a small set of annotated manuscripts and evaluate how well the model was able to annotate each word in a held-out portion of the dataset. Then we turn to evaluating the model in the context of ranked retrieval.

The data set we used in training and evaluating our approach consists of 20 manually annotated pages from George Washington's handwritten letters. Segmenting this collection yielded a total of 4773 images, from which the majority contain exactly one word. An estimated 5-10% of the images contain segmentation errors of varying degrees: parts of words that have faded tend to get missed by the segmentation, and occasionally images contain 2 or more words or only a word fragment.

5.4.3.1 Evaluation methodology

Our dataset comprises 4773 total word occurrences arranged on 657 lines. Because of the relatively small size of the dataset, all of our experiments use a 10-fold randomized cross-validation, where each time the data is split into a 90% training and 10% testing sets. Splitting was performed on a line level, since we chose lines to be our retrieval unit. Prior to any experiments, the manual annotations were reduced to the root form using the Krovetz [3] morphological analyzer. This is a standard practice in information retrieval, it allows one to search for semantically similar variants of the same word. For our annotation experiments we use every word of the 4773-word vocabulary that occurs in both the training and the testing set. For retrieval experiments, we remove all function words, such as "of", "the", "and", etc. Furthermore, to simulate real queries users might pose to our system, we tested all possible combinations of 2, 3 and 4 words that occurred on the same line in the testing, but not necessarily in the training set. Function words were excluded from all of these combinations.

We use the standard evaluation methodology of information retrieval. In response to a given query, our model produces a ranking of all lines in the testing set. Out of these lines we consider only the ones that

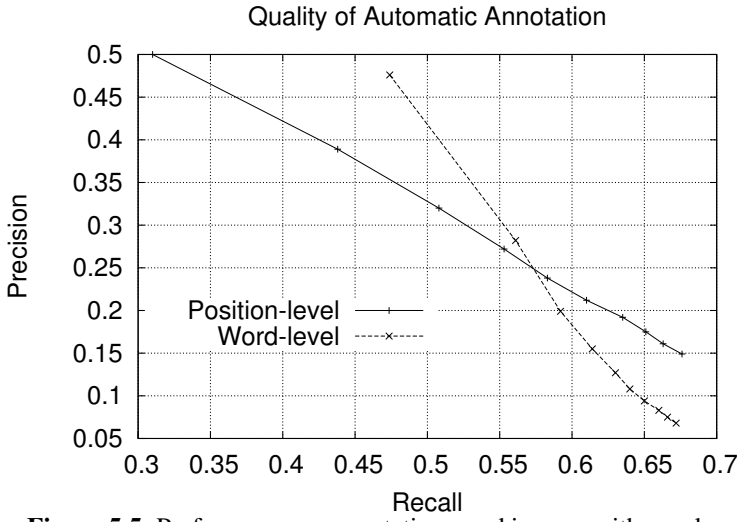


Figure 5.5. Performance on annotating word images with words.

contain all query words to be relevant. The remaining lines are assumed to be non-relevant. Then for each line in the ranked list we compute *recall* and *precision*. Recall is defined as the number of relevant lines above (and including) the current line, divided by the total number of relevant lines for the current query. Similarly, precision is defined as number of above relevant lines divided by the rank of the current line. Recall is a measure of what percent of relevant lines we found, and precision suggests how many non-relevant lines we had to look at to achieve that recall. In our evaluation we use plots of precision vs. recall, averaged over all queries and all cross-validation repeats. We also report Mean Average Precision, which is an average of precision values at all recall points.

5.4.3.2 Results: annotating images with words

Figure 5.5 shows the performance of our model on the task of assigning word labels to handwritten images. We carried out two types of evaluation. In **position-level** evaluation, we generated a probability distribution $E_{\mathbf{d}}\theta(\cdot)$ for every image $\mathbf{d}$ in the testing set. Then we looked for the rank of the correct word in that distribution and averaged the resulting recall and precision over all positions. Since we did not exclude function words at this stage, position-level evaluation is strongly biased toward very common words such as "of", "the" etc. These words are generally not very interesting, so we carried out a **word-level** evaluation. Here for a given word e we look at the ranked list of all the individual word images $\mathbf{d}$ in the testing set, sorted in the decreasing order of $E_{\mathbf{d}}\theta(e)$. This is similar to running e as a query and retrieving all *positions* in which it could possibly occur. Recall and precision were calculated as discussed in the previous section.

From the graphs in Figure 5.5 we observe that our model performs quite well in annotation. For position-level annotation, we achieve 50% precision at rank 1, which means that for a given image $\mathbf{d}$, half the time the word e with the highest conditional probability $E_{\mathbf{d}}\theta(e)$ is the correct one. Word-oriented evaluation also has close to 50% precision at rank 1, meaning that for a given word e the highest-ranked image $\mathbf{d}$ contains that word almost half the time. Mean Average Precision values are 54% and 52% for position-oriented and word-oriented evaluations respectively.

5.4.3.3 Results: retrieving images with a text query

Now we turn our attention to using our model for the task of retrieving relevant portions of manuscripts. As discussed before, we created four sets of queries: 1, 2, 3 and 4 words in length, and tested them on retrieving line segments. Our experiments involve a total of 1950 single-word queries, 1939 word pairs, 1870 3-word and 1558 4-word queries over 657 lines. Figure 5.6 shows the recall-precision graphs. It is very encouraging to see that our model performs extremely well in this evaluation, reaching over 90% mean precision at rank 1.

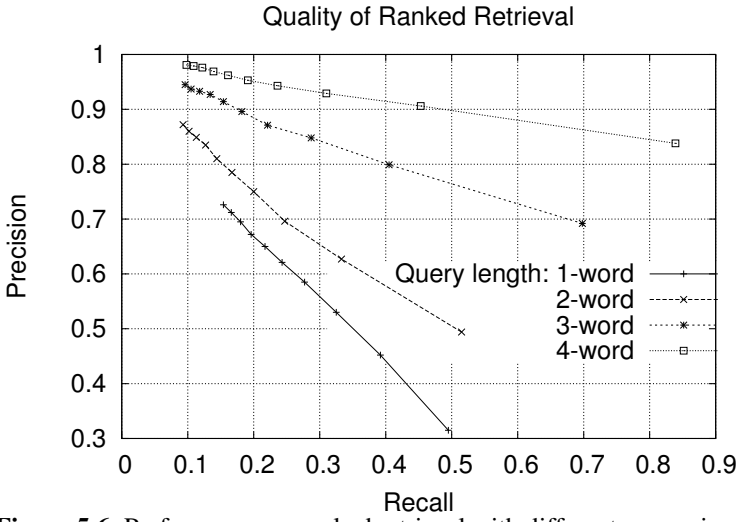


Figure 5.6. Performance on ranked retrieval with different query sizes.

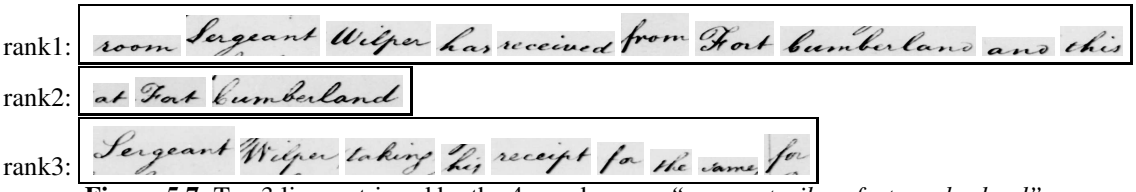


Figure 5.7. Top 3 lines retrieved by the 4-word query: *"sergeant wilper fort cumberland"*.

This is an exceptionally good result, showing that our model is nearly flawless when even such short queries are used. Mean average precision values were 54%, 63%, 78% and 89% for 1-, 2-, 3- and 4-word queries respectively. Figure 5.7 shows three top-ranked lines returned in response to a 4-word query *"sergeant wilper fort cumberland"*. Only one line in the entire collection is relevant to the query; the line is ranked at the top of the list.

5.5 Image Retrieval

We will discuss the problem of automatic annotation of large collections of images. This task is now becoming particularly pertinent due to a combination of three factors: rise in the popularity and ease of digital photography, increased availability of high-bandwidth connections and the ever growing storage capacity. A typical personal computer nowadays stores an astounding number of images. These may include photographs of family and friends, greeting cards, clip-art collections, diagrams and a wide variety of other images. The web, e-mail and peer-to-peer networks make it incredibly easy to share these images with wide audiences, leading to larger and larger personal collections. Unfortunately, as these collections grow they become harder and harder to organize and search through. Most of the images in personal collections are not annotated in a meaningful way: in the best case the file name contains a single keyword indicative of the contents, in the worst case images are organized by date. Finding an image in a collection like this is quite challenging, as it involves browsing through a large portion of them. An alternative is to label them manually, but that is a tedious endeavor that few users are willing to consider.

In this section we discuss how we could do the labeling automatically. Our discussion will build upon previous results. In the previous section we demonstrated that our generative model is able to successfully associate English words with images showing handwritten expressions of these words. However, nothing in our model is specific to handwriting, and in this section we will show that the generative model is capable of learning relations between English words and images of general objects.

That being said, we have to admit that general images are somewhat more complex than handwritten words. In the handwriting scenario there was only one correct word for each image. We cannot assume the same about photographs or other general images. Any realistic image is going to contain a large number of objects, some perhaps more prominent than others. An accurate description of all the objects present in a given image may require a rather long narrative.

5.5.1 Representation

We have a collection $\mathcal{C}$ of un-annotated test images and another collection $\mathcal{C}_{train}$ of labeled images. Our immediate goal is use the training collection to come up with plausible labels for the test images. The ultimate goal is being able to retrieve unlabeled images by issuing text queries. Note that general images are somewhat more difficult than handwritten words.

We will use the following representation space in this scenario. We assume that the latent form X of each item in the collection consists of two parts. The first part is a bitmap $\mathbf{d}$. The second is a complete narrative $\mathbf{w}$ detailing the contents of the bitmap. The query transform will operate in the same way as in the ad-hoc scenario (section 5.1.1): Q will take the latent representation $\{\mathbf{d}, \mathbf{w}\}$, discard the bitmap $\mathbf{d}$ and then return out a short sub-sequence $w_1 \ldots w_m$ as the query. Which words from $\mathbf{w}$ are selected as keywords will be determined by the same heuristic we discussed in section 5.1.1.

The document generating transform D is more complex. Similar to the handwriting scenario, we will try to express the bitmap in terms of a set of feature functions. But now, since we are dealing with general images, we have to compensate for the fact that an image may contain multiple objects, and each object will be associated with a distinct type of vocabulary. The transform will consist of three distinct steps. The first step of D will be to partition the image into a set of regions $\mathbf{d}_1 \ldots \mathbf{d}_k$, each containing a distinct object. The partitioning algorithm can be as simple as slicing the bitmap according to a rectangular grid. Or it can be a complex procedure like *blob-world* [22] or *normalized cuts* [120]; both attempt to find object boundaries based on local inconsistencies among the pixels.

The second step of the transform is to compute a set of representative features for each region of the bitmap. A popular set of features is suggested by [41]. Their feature set is dominated by color statistics for the region, but also includes relative position of the regions in the overall bitmap, a few shape-related features, such as area and circumference, and a number of texture statistics computed by applying Gabor filters to the region. The feature set has been used in a number of publications attempting to solve the image annotation problem. After the second step, a bitmap $\mathbf{d}$ is represented by a set of l-dimensional real-valued vectors $\vec{r}_1 \ldots \vec{r}_k$, one for each region in the bitmap.

The third step of the document transform D is optional. If we were to mimic the handwriting scenario, we would converting the real-valued feature vectors $\vec{r}_1 \ldots \vec{r}_k$ into discrete features $f_1 \ldots f_k$ taking values in some feature vocabulary $\mathcal{F}$. One possible way to discretize the real-valued vectors is to *cluster* them, as was done by [41, 63]. After the clustering, each feature vector $\vec{r}_i$ is replaced by the number of the cluster it falls into. In this case, the feature vocabulary $\mathcal{F}$ is simply the set of cluster numbers $\{1 \ldots K\}$. After the third step, we find our representation to be identical to the representation used in the handwriting scenario: we have a set of English words $w_1 \ldots w_n$ and a set of feature words $f_1 \ldots f_k$. In this case components of the model will be defined exactly as in section5.4.2.2; document ranking will be done identically to section 5.4.2.3. The model constructed in this fashion will be referred to as *CMRM* in the experiments.

5.5.1.1 Problems arising from clustering

The process of discretization described above involves clustering, which is a dimensionality reduction technique. As we argued in sections 4.2.7 and 4.4.3, any sort of dimensionality reduction will lead to very undesirable effects: we run a very real risk of wiping out the rare events that our user may be trying to find. For example, suppose that our database consists primarily of animal photographs, but also contains one or two technical diagrams. A typical factored model, whether it is based on clustering, mixture models or principal component analysis is very unlikely to assign a separate cluster / component to the diagrams. In all likelihood the diagrams will be folded into the general structure, i.e. collapsed into the closest animal cluster or represented as a mixture of fish, birds and insects. This is not a problem if our user is interested in animals. But it will pose a serious difficulty if he or she is seeking technical drawings.

5.5.1.2 Continuous-space generative model

As a remedy to the above problem, we could choose to skip the discretization process entirely. Instead, we could represent the bitmap by the real-valued feature vectors $\vec{r}_1 \ldots \vec{r}_k$, and try to correlate these vectors with annotation words $w_1 \ldots w_m$. The advantage of not clustering is obvious: we should be able to handle events of much finer granularity. The disadvantage is that we have to re-formulate certain parts of our representation to handle real-valued random variables. The annotation words $w_1 \ldots w_m$ will be modeled by multinomial distributions $u : \mathcal{V} \mapsto [0,1]$. The feature vectors $\vec{r}_1 \ldots \vec{r}_k$ are elements of $\mathbb{R}^d$, where d is the number of features we compute for each region. These vectors will be modeled by a probability density $h : \mathbb{R}^d \mapsto [0,1]$. Accordingly, the parameter space Θ will be the cross product $\mathbb{P}_\mathcal{V} \times \mathbb{F}_d$, where $\mathbb{P}_\mathcal{V}$ is the vocabulary simplex and $\mathbb{F}_d$ is the space of all probability density functions over $\mathbb{R}^d$. We will continue to use kernel-based density allocation, leading to the following expression for the likelihood of observing image features $\vec{r}_1 \ldots \vec{r}_k$ together with annotation $w_1 \ldots w_m$:

$$\begin{aligned} P_\mathcal{R}(w_1 \ldots w_m, \vec{r}_1 \ldots \vec{r}_k) &= \int_{\mathbb{P}_\mathcal{V}} \int_{\mathcal{F}_d} \left(\prod_{i=1}^{m} u(w_i) \right) \left(\prod_{i=1}^{k} h(\vec{r}_i) \right) p_{ker,\delta}(du \times dh) \\ &= \frac{1}{N} \sum_{\mathbf{x}} \left(\prod_{i=1}^{m} u_\mathbf{x}(w_i) \right) \left(\prod_{i=1}^{k} h_\mathbf{x}(\vec{r}_i) \right) \end{aligned} \tag{5.25}$$

The summation goes over all training examples $\mathbf{x} \in \mathcal{C}_{train}$.

5.5.1.3 Parameter estimation for the continuous model

As in the previous cases, all parameters in the model are tied to the individual training examples. We assume that our training collection $\mathcal{C}_{train}$ contains a set of annotated images. For each example $\mathbf{x} = \{\mathbf{d}, \mathbf{w}\}$ in the training set we need to define two components: the multinomial distribution $u_\mathbf{x}$ and the probability density $h_\mathbf{x}$. The multinomial $u_\mathbf{x}$ reflects the empirical distribution of words in the annotation string $\mathbf{w}$. We define this distribution in the same way as for all previous examples:

$$u_\mathbf{x}(v) = \lambda \frac{n(v,\mathbf{w})}{|\mathbf{w}|} + (1-\lambda) \frac{\sum_\mathbf{w} n(v,\mathbf{w})}{\sum_\mathbf{w} |\mathbf{w}|} \tag{5.26}$$

As usual, $n(v,\mathbf{w})$ stands for the number of times we observe the word v in the annotation $\mathbf{w}$, the quantity $|\mathbf{w}|$ represents annotation length and the summation goes over all training annotations.

The function $h_\mathbf{x}$ is a probability density function over the feature space $\mathbb{R}^d$. We will use a kernel-based estimate to model $h_\mathbf{x}$. Recall that the bitmap portion $\mathbf{d}$ of each training example is represented by a set of k feature vectors $\vec{r}_1 \ldots \vec{r}_k$. We will place a d-dimensional Gaussian kernel on top of each vector $\vec{r}_j$, leading to the following probability density estimate at a given point $\vec{s} \in \mathbb{R}^d$:

$$h_\mathbf{x}(\vec{s}) = \frac{1}{k} \sum_{j=1}^{k} \frac{1}{\sqrt{2^d \pi^d \beta^d}} \exp\left(\frac{||\vec{s} - \vec{r}_j||}{\beta} \right) \tag{5.27}$$

Here $||\vec{s} - \vec{r}_j||$ is the Euclidean distance between feature vectors $\vec{s}$ and $\vec{r}_j$. Parameter β is the kernel *bandwidth*, it determines whether the kernel is highly peaked around its center point $\vec{r}_j$, or whether it is spread out.

5.5.1.4 Document ranking

Image annotation and ranked retrieval will be carried out in the same manner as in the handwriting scenario. For each unlabeled testing image $\mathbf{d}$ with features $\vec{r}_1 \ldots \vec{r}_k$ we will construct a posterior language model over the vocabulary:

$$E_\mathbf{d} u(v) = \frac{\int_{\mathbb{P}_\mathcal{V}} \int_{\mathcal{F}_d} u(v) \left\{ \prod_{i=1}^{k} h(\vec{r}_i) \right\} p_{ker,\delta}(du \times dh)}{\int_{\mathbb{P}_\mathcal{V}} \int_{\mathcal{F}_d} \left\{ \prod_{i=1}^{k} h(\vec{r}_i) \right\} p_{ker,\delta}(du \times dh)} = \frac{\sum_\mathbf{x} u_\mathbf{x}(v) \prod_{i=1}^{k} h_\mathbf{x}(\vec{r}_i)}{\sum_\mathbf{x} \prod_{i=1}^{k} h_\mathbf{x}(\vec{r}_i)} \tag{5.28}$$

We will then use the language model $E_{\mathbf{d}}u(v)$ to assign most probable annotation words v to the testing image. We will also use $E_{\mathbf{d}}u(\cdot)$ to assign probabilities to multi-word queries, giving us a way to retrieve unlabeled images in response to the textual query $q_1 \ldots q_m$:

$$P(q_1 \ldots q_m) = \prod_{j=1}^{m} E_{\mathbf{d}}u(q_j) \tag{5.29}$$

The continuous model described in the present section will be referred to as *CRM* in our experiments.

5.5.2 Experiments

In this section we provide a thorough evaluation of our model on two real-world tasks mentioned above. First we test the ability of our model to assign meaningful annotations to images in a held-out testing set. Then, in section 5.5.2.3 we evaluate the ability of our model to retrieve un-annotated images using text-only queries. In both cases we compare performance of our models to established baselines and show significant improvements over the current state-of-the-art models. Experiments discussed in this section were originally reported in [63] and [75].

5.5.2.1 Experimental Setup

To provide a meaningful comparison with previously-reported results, we use, without any modification, the dataset provided by Duygulu et al.[41][2]. This allows us to compare the performance of models in a strictly controlled manner. The dataset consists of 5,000 images from 50 Corel Stock Photo CDs. Each cd includes 100 images on the same topic. Each image contains an annotation of 1-5 keywords. Overall there are 371 words. Prior to modeling, every image in the dataset is pre-segmented into regions using general-purpose algorithms, such as normalized cuts [120]. We use pre-computed feature vectors for every segmented region r. The feature set consists of 36 features: 18 color features, 12 texture features and 6 shape features. For details of the features refer to [41]. We divided the dataset into 3 parts - with 4,000 training set images, 500 evaluation set images and 500 images in the test set. The evaluation set is used to find the smoothing meta-parameters β and λ. After fixing the parameters, we merged the 4,000 training set and 500 evaluation set images to make a new training set. This corresponds to the training set of 4500 images and the test set of 500 images used by Duygulu et al.[41].

5.5.2.2 Results: Automatic Image Annotation

In this section we evaluate the performance of our model on the task of automatic image annotation. We are given an un-annotated image $\mathbf{d}$ and are asked to automatically produce an annotation. The automatic annotation is then compared to the held-out human annotation $\mathbf{w}$. We follow the experimental methodology used by[41, 63]. Given a set of image features $\vec{r}_1 \ldots \vec{r}_k$ we use equation (5.28) to arrive at the posterior language model $E_{\mathbf{d}}u(\cdot)$. We take the top 5 words from that distribution and call them the automatic annotation of the image in question. Then, following [41], we compute annotation recall and precision for every word in the testing set. Recall is the number of images correctly annotated with a given word, divided by the number of images that have that word in the human annotation. Precision is the number of correctly annotated images divided by the total number of images annotated with that particular word (correctly or not). Recall and precision values are averaged over the set of testing words.

We compare the annotation performance of the four models: the Co-occurrence Model [90], the Translation Model [41], CMRM and the CRM. We report the results on two sets of words: the subset of *49 best* words which was used by[41, 63], and the complete set of all 260 words that occur in the testing set. Table 5.11 shows the performance on both word sets. The figures clearly show that the continuous-space generative model (CRM) substantially outperforms the other models and is the only one of the four capable of producing reasonable mean recall and mean precision numbers when every word in the test set is used. In Figure5.8 we provide sample annotations for the two best models in the table, CMRM and CRM, showing that the CRM is considerably more accurate.

[2] Available at http://www.cs.arizona.edu/people/kobus/ research/data/eccv_2002

Models	Co-occurrence	Translation	CMRM	CRM
#words with recall ≥ 0	19	49	66	107 +62%
Results on 49 best words, as in[63, 8]				
Mean per-word Recall	-	0.34	0.48	0.70 +46%
Mean per-word Precision	-	0.20	0.40	0.59 +48%
Results on all 260 words				
Mean per-word Recall	0.02	0.04	0.09	0.19 +111%
Mean per-word Precision	0.03	0.06	0.10	0.16 +60 %

Table 5.11. Comparing recall and precision of the four models on the task of automatic image annotation. CRM substantially outperforms all other models. Percent improvements are over the second-best model (CMRM). Both CRM and CMRM significantly outperform the state-of-the-art translation model.

Images				
CMRM Annotation	water sky plane bear	water sky plane jet tree	water sky tree people	people rocks water buildings
CRM Annotation	lizard marine iguana rocks	snow bear polar tundra	train railroad tracks locomotive	cat tiger water forest

Figure 5.8. The generative model based on continuous features (CRM) performs substantially better than the discrete cross-media relevance model (CMRM) for annotating images in the test set.

Figure 5.9. Example: top 5 images retrieved by CRM in response to text query *"cars track"*

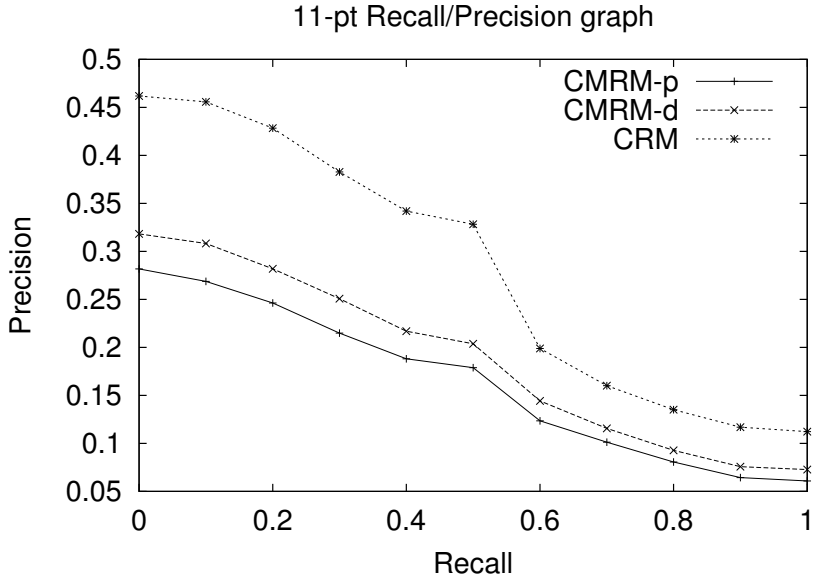


Figure 5.10. Recall-precision curves showing retrieval performance of the models with 2-word queries. The model using real-valued feature vectors (CRM) substantially outperforms the two flavors of the discrete model (CMRM).

Query length	1 word	2 words	3 words	4 words
Number of queries	179	386	178	24
Relevant images	1675	1647	542	67
Precision after 5 retrieved images				
CMRM	0.1989	0.1306	0.1494	0.2083
CRM	0.2480 **+25%**	0.1902 **+45%**	0.1888 **+26%**	0.2333 +12%
Mean Average Precision				
CMRM	0.1697	0.1642	0.2030	0.2765
CRM	0.2353 **+39%**	0.2534 **+54%**	0.3152 **+55%**	0.4471 +61%

Table 5.12. Comparing CRM and CMRM on the task of image retrieval. Our model outperforms the CMRM model by a wide margin on all query sets. Boldface figures mark improvements that are statistically significant according to sign test with a confidence of 99% (p-value < 0.01).

5.5.2.3 Results: Ranked Retrieval of Images

In this section we turn our attention to the problem of ranked retrieval of images. In the retrieval setting we are given a text query $q_1 \ldots q_m$ and a testing collection of un-annotated images. For each testing image $\mathbf{d}$ we use equation (5.28) to get the conditional distribution $E_{\mathbf{d}}u(\cdot)$. All images in the collection are ranked according to the query likelihood criterion under $E_{\mathbf{d}}u(\cdot)$. We use four sets of queries, constructed from all 1-, 2-, 3- and 4-word combinations of words that occur at least twice in the testing set. An image is considered relevant to a given query if its *manual* annotation contains all of the query words. As our evaluation metrics we use precision at 5 retrieved images and non-interpolated average precision[3], averaged over the entire query set. Precision at 5 documents is a good measure of performance for a casual user who is interested in retrieving a couple of relevant items without looking at too much junk. Average precision is more appropriate for a professional user who wants to find a large proportion of relevant items.

Table 5.12 shows the performance of CRM on the four query sets, contrasted with performance of the CMRM baseline on the same data. We observe that our model substantially outperforms the CMRM baseline

[3] Average precision is the average of precision values at the ranks where relevant items occur.

on every query set. Improvements in average precision are particularly impressive, our model outperforms the baseline by 40 - 60 percent. All improvements on 1-, 2- and 3-word queries are statistically significant based on a sign test with a p-value of 0.01. We are also very encouraged by the precision our model shows at 5 retrieved images: precision values around 0.2 suggest that an average query always has a relevant image in the top 5. Figure 5.9 shows top 5 images retrieved in response to the text query *"cars track"*.

5.6 Video Retrieval

Collections of video footage represent a mesmerizing source of information. Personal video collections remain relatively rare to this day, partly because digitizing video is more time consuming than digitizing pictures. However, there exist countless public archives of digitized video. These archives contain anything from news reports, to movies, to historical footage, to recordings made by security cameras. A particularly fascinating example is the collection assembled by the Shoah Visual History Foundation[4]. Established by Steven Spielberg, the archive is truly immense. It contains hundreds of thousands of hours of testimony from nearly 52,000 survivors of the Holocaust. Searching through a video archive of this size is a significant challenge. Video footage has no innate annotations, so finding a certain segment typically involves the user rewinding and fast-forwarding until he sees what he likes. This is feasible on a relatively small scale, but completely out of the question if we are dealing with 100,000 hours of potentially relevant video. A typical solution is hand-labeling the tapes with meta-data. For example, the Shoah Foundation Archive provides extensive biographical information for each interviewee; it includes gender, place and date of birth, religious identity, languages spoken and a lot of other information. However, all this information cannot help us if we are looking for specific content on the tape. For example, we have no way of searching for segments that contain something other than the interviewee: such as photographs, letters or other articles that may have been filmed during the testimony.

One possible solution to the video retrieval problem is to use Automatic Speech Recognition on the audio accompanying the footage. However, this assumes that **(i)** everything shown in the video is appropriately narrated and **(ii)** we have a highly accurate speech recognition system for the language in which the narrative was given. The second requirement is not trivial, for instance the Shoah Archive contains testimonies in 31 languages, ranging from English and German to Croatian and Romani. Finding a speech recognizer that is able to handle Romani may turn out to be a challenge.

In this section we will describe our solution to the video retrieval problem. The solution involves *annotating* individual keyframes in the video with appropriate keywords. The keywords will describe the objects contained in the frame. These annotations will allow us to search through the video by issuing text queries. An important advantage of this approach is that it is language-independent; an airplane in the video footage is going to look like an airplane regardless of the narration language. Consequently, our approach will allow us to issue an English query, e.g. *"riots"*, and retrieve relevant video footage from Arabic, Chinese or Russian video collections or streams.

5.6.1 Representation

Our video retrieval model will closely follow the image retrieval model described in the previous chapter. A video archive is a collection of keyframes automatically extracted from video. Each keyframe is a bitmap image, no different from the images discussed in the previous chapter. We hypothesize that in its latent form each keyframe is coupled with a long English narrative describing the objects contained in the keyframe. The representation will be identical to the image retrieval scenario (see section 5.5.1). We will segment the keyframes into a set of regions and compute the feature vectors $\vec{r}_1 \ldots \vec{r}_k$ for each region. Given the superior performance of the continuous-space model (CRM), we will not discretize the features. The generative relevance model will take the form of equation (5.25). Parameters will be estimated in exactly the same manner as suggested in section 5.5.1.3.

There will be two minor differences in application of our model to the video domain. First, instead of using sophisticated segmentation algorithms like blob-world [22] and normalized-cuts [120], we will slice up each keyframe according to a rectangular grid. There are two motivations for this step: (i) the computational

[4] http://www.vhf.org

expense of segmentation becomes a serious problem for video and (ii) there are some indications [44, 74] that rectangular regions perform better than more sophisticated segments.

The second difference from the previous scenario will be in the way we pick the smoothing parameter λ in equation (5.26). Previously, we treated λ as a constant independent of the annotation length $|\mathbf{w}|$. The value of the constant was determined by optimizing retrieval performance on the development set. In the current scenario we will set λ to be the fraction $\frac{|\mathbf{w}|}{K}$, where $|\mathbf{w}|$ is the annotation length of the example and K is a large constant, bigger than the maximum observed annotation length $\max_{\mathbf{w}}\{|\mathbf{w}|\}$. The effect of setting λ in this fashion will be to compensate for the varying lengths of training annotations. This is necessary because of a somewhat erratic nature of example annotations in our video collection. The same frame showing a reporter can be annotated either with a single word [*"face"*], or with a set of words [*"face", "male_face", "reporter", "indoors", "studio_setting"*]. When we keep λ constant, the first annotation would lead to a maximum likelihood probability of 1 for the word *"face"*. The second annotation would give the same word a probability of $\frac{1}{5}$. By setting $\lambda = \frac{|\mathbf{w}|}{K}$ we make the probability estimates comparable under the two examples: the probability of *"face"* in both cases will be $\frac{1}{K}$. The effect is that all annotations look like they have the same length; we will refer to this model as the *fixed-length CRM*.

5.6.2 Experiments

Experiments discussed in this section were originally reported in [44, 74]. We provide the experimental results of the retrieval task over a keyframe dataset, which is a subset of the Video Track dataset of the TREC conference. The data set consists of 12 MPEG files, each of which is a 30-minutes video section of CNN or ABC news and advertisements. 5200 key frames were extracted and provided by NIST for this dataset. The participants in TREC annotated a portion of the videos. The word vocabulary for human annotation is represented as a hierarchical tree with each annotation word as a node, which means many key frames are annotated hierarchically, e.g. a key frame can be assigned a set of words like "face, female_face, female_news_person". The annotation length for key frames can vary widely. There are 137 keywords in the whole dataset after we ignore all the audio annotations.We randomly divide the dataset into a training set (1735 key frames), a validation set (1735 key frames) and a test set (1730 key frames). The validation set is used to find system parameters, and then merged into the training set after we find the parameters.

Every keyframe in this set is partitioned into rectangular grids, and a feature vector is then calculated for every grid region. The number of rectangles is empirically selected (using the training and validation sets), it is 35 for each sample. The feature set consists of 30 features: 18 color features (including region color average, standard deviation and skewness) and 12 texture features (Gabor energy computed over 3 scales and 4 orientations).

In our retrieval experiments, we use three set of queries, constructed from all 1-, 2-, and 3-word combinations that occur at least 10 times in the testing set. For each set of queries, we perform the comparative experiments over the two different methods for setting the smoothing parameter λ. An image is considered relevant to a given query if its manual annotation contains all the query words. As used in [44, 74, 75], evaluation metrics are precision at 5 retrieved keyframes and non-interpolated average precision, averaged over the entire query set. These two different metrics are good measures suitable for distinct needs of casual users and professional users.

Table 5.13 shows the details of the performance of our two different methods over the three sets of queries. We can observe that fixed-length CRM substantially outperforms the original formulation. The improvements are 15%, 37% and 33% on the 1-, 2- and 3-word query sets respectively. The precision at 5 retrieved keyframes also indicates that for fixed-length CRM, there are half images relevant to the query in the top 5. Figures 5.11 and 5.12 show the top 4 images in the ranked lists corresponding to the text queries *"basketball"* and *"outdoors sky transportation"* respectively. In both figures, the top row represents performance of the regular CRM, the bottom row shows the results from fixed-length CRM.

5.7 Topic Detection and Tracking

All previously considered tasks were fairly intuitive – we did not need to spend much time motivating the ad-hoc retrieval process or the video searching scenario. Our final scenario is slightly different, and without

Query length	1 word	2 words	3 words
Number of queries	107	431	402
Relevant images	6649	12553	11023
Mean precision after 5 retrieved keyframes			
CRM	0.36	0.33	0.42
CRM (fixed length)	0.49	0.47	0.58
Mean Average Precision			
CRM	0.26	0.19	0.25
CRM (fixed-length)	0.30	0.26	0.32

Table 5.13. Performance of CRM on the TREC video retrieval task.

Figure 5.11. First 4 ranked results for the query *"basketball"*. Top row: retrieved by CRM. Bottom row: retrieved by fixed-length CRM.

Figure 5.12. First 4 ranked results for the query *"outdoors, sky, transportation"*. Top row: CRM results. Bottom row: fixed-length CRM results.

some introduction it may appear somewhat artificial and confusing. In addition, topic detection does not really fit a strict definition of a retrieval scenario, where we typically have a user, her query and a static collection of documents. To provide a smooth transition from previous scenarios, we will spend a little bit of time introducing the big picture of the problem we are dealing with. A much more detailed exposition of Topic Detection and Tracking can be found in [1, 5, 100] and in the proceedings of the annual TDT workshop organized by NIST.

5.7.1 Definition

Topic Detection and Tracking is a research program dedicated to event-based organization of news reports. The idea is to take live news reports coming from newswire, television and radio sources, and organize them in a way that is intuitive and convenient to users. The focus of TDT is on dealing with live news, not with static collections. There are a number of distinct characteristics that make TDT very different from the tasks we previously described.

If we were to pick a single quality that makes TDT unusual, it would be the concept of *relevance* that is used by the program participants. Compared to other areas of information retrieval, TDT has an unusually clear and crisp definition of this fuzzy concept. The definition is based on a notion of *event*. An event is something that happens at a specific place and time and involves specific participants, be they human or otherwise. For example "the 1998 landing of hurricane Mitch" represents a specific event, whereas "deadly hurricanes" does not. A TDT *topic* embodies all activities directly related to a given *event*.

The term relevance is never used by the TDT program participants, but if it were used, it would mean that a particular story discusses the event in question, or discusses some activity directly related to that event. How does this relevance fit into the definitions of relevance discussed in chapter 2? By and large, it still fits into the bounds we delineated for this thesis. TDT relevance is concerned with topicality, rather than usefulness to some user task. TDT is non-interactive, any user feedback must happen before the TDT system starts operating. TDT deals with full-text documents, and as we will see these documents are quite varied in their nature. On the other hand, there are no queries in TDT, and this makes the problem quite interesting. In TDT, an information need is never expressed as a request, it can appear only in three possible forms: **(i)** it can be exemplified by a small set of on-topic stories (tracking task), **(ii)** it exists only as an un-observed variable, (clustering and link detection tasks), and **(iii)** it is *topical novelty*, i.e. the user is interested in things that have never been previously reported (new event detection task).

Another way in which TDT is dramatically different from other retrieval scenarios is the type of user for which the technology is targeted. A typical retrieval scenario, such as ad-hoc retrieval or video search, will target a casual user who will not expend a lot of effort formulating his request, and will likely be satisfied with a very small number of relevant hits. In TDT the target user is quite different. He or she is a professional analyst interested in seeing *all* news reports related to a particular event. One could say that the target TDT user is much more recall-oriented than a typical web user. Because of this difference, TDT evaluation metrics are very different from the more traditional precision and recall.

TDT systems are supposed to operate on live news feeds, and that presents additional challenges of algorithmic design. In all previous scenarios we were dealing with a static collection of documents. The main advantage of a static collection is that we can learn a statistical model once, and then use it for all queries issued against that collection. In TDT we do not have the same luxury. Whatever model we come up with has to grow and adapt to match the ever-changing nature of a live news-feed. Fortunately for us, adaptation and expansion is straightforward with the kernel-based models we proposed in the previous chapter. The on-line nature of TDT leads to another complication. In all previous scenarios it was sufficient to *rank* the documents in our collection, and the user himself would decide when it is time to stop examining documents. In TDT we cannot afford the same process. The user needs to be alerted to events of interest as soon as possible, so for every document in the stream we need to make a hard decision of whether this document is relevant or not.

Last but not least, a TDT system has to deal successfully with very heterogeneous streams of data. Existing TDT collections represent a mix of newswire reports (e.g. Associated Press), editorial articles (e.g. New York Times), radio broadcasts (Voice of America) and TV news shows (CNN, ABC, NileTV). Newswire and editorials come in a nicely delineated textual form, but radio and TV shows come as a continuous stream of audio. TDT participants have access to a textual transcript of the audio, either as closed captions or as the

output of the Automatic Speech Recognition system (ASR). The quality of transcribed text is substantially lower than the quality of newswires: for example names and locations are often garbled or spelled incorrectly. TDT participants have an option of working with the audio signal instead of the transcript. Another complication of TDT is its multi-lingual nature. The program started by focusing exclusively on English news, but over the years grew to include Chinese and Arabic sources. To make the task particularly challenging, TDT specifically focuses on events that receive news coverage in all three languages.

The TDT initiative defines five tasks, each of which represents one aspect of a complete news processing system. These tasks are:

1. **Story Segmentation.** Live news reporting comes in a form of a continuous stream of text. It is not broken down into cohesive units that we are used to working with. News anchors typically give very little indication that they are switching from one story to the next. This is not problematic for humans, who are naturally good at handling rapid semantic shifts, but it does pose a formidable challenge to automatic systems. The first task of TDT is to take a continuous stream of text and break it down into meaningful chunks, where each chunk discusses a particular topic, distinctly different from the preceding and the following chunks. These chunks are called *stories* and form the basic processing unit for the other four tasks.

2. **New Event Detection.** The purpose of this task (NED) is to identify the first reporting of some new event. For example, when an earthquake strikes, we want to flag the very first report of that event as *new*. Every subsequent report discussing the same earthquake would be marked as *old*. New Event Detection is a very interesting problem, in a way it directly addresses the problem of redundancy we discussed in section 2.1.5.1. The problem turns out to be very challenging: NED error rates remain high despite years of dedicated efforts by the researchers. A possible reason for that is that NED has no scope: it provides no intuition for what we *should* look for in a report; the only thing we know is what we *should not* look for: we should not retrieve anything we have seen before. Allan, Lavrenko and Jin [7] presented a formal argument showing that the New Event Detection problem cannot be solved using existing methods.

3. **Topic Tracking.** In the tracking task we start with a small set of stories discussing a particular event; the goal is to find all subsequent reports addressing the same event. Tracking is a step that naturally follows NED: once we have spotted some new interesting event, we want to track it in the news. The tracking task is the only TDT task where we are given explicit examples of the topic we are looking for. It is similar to information filtering or routing. The main differences are: (i) the number of training stories is very small: typically 1 to 4, (ii) we need to track a specific *event*, not the general *subject*, and (iii) we need to track in multiple languages and across multiple media.

4. **Topic Detection.** This task is also known as the *clustering* task, because the goal is to organize the news stream into a set of clusters, such that every cluster would contain stories discussing some particular event. What makes the detection task very different from a normal clustering task is that we have to operate in a live, on-line environment. As soon as a story comes off the wire we have to either assign it to one of the existing clusters or create a new cluster if the story addresses a new, previously unseen event. The on-line nature of the task automatically rules out the use of a large number of popular clustering algorithms: such as top-down partitioning techniques, or agglomerative methods. Another way to look at the detection task is as a combination of the NED and the tracking tasks: first we detect a new event in the news stream, then we track the event using the first story as a single training example.

5. **Link Detection.** The last TDT task appears extremely simple and does not have the intuitive appeal of the previous tasks. We are given two stories and asked to decide whether or not they discuss the same event. We get no information regarding the type of event. The stories can come from different sources (e.g. newswire and radio), and can be written in two different languages. Despite its apparent simplicity, the link detection task is a very important part of TDT. We can view it as a component technology that is required by every other task. It is fairly obvious that a perfect solution to the link detection task would automatically solve all other TDT tasks. But we can make an even stronger claim:

in our opinion, it is not possible to improve the current performance of NED, tracking and detection systems, unless we substantially improve the accuracy of link detection.[5]

5.7.2 Representation

In light of its fundamental role, we are going to focus all our TDT experiments on the Link Detection task. The task has the following setup. We are given two stories, $\mathbf{d}^1$ and $\mathbf{d}^2$; our goal is to decide whether the two stories discuss the same topic. The stories are selected randomly from the news stream; without loss of generality we will assume that $\mathbf{d}^2$ was released after $\mathbf{d}^1$. We have a collection of documents $\mathcal{C}$, which represents the news stream up to $\mathbf{d}^2$. For simplicity, we will assume that all documents in the collection are written in the same language. When they are not, we assume that original documents are mapped to some common language using a statistical dictionary (see section 5.3.1.4 for details).

The details will closely follow the relevance feedback scenario (section 5.2). The latent representation X of each story consists of a string length, followed by $M-1$ words from the common vocabulary $\mathcal{V}$. The document generating transform will operate exactly as in the ad-hoc case (equation 5.1); it will return the first X_1 words from X. There are no queries in TDT, so the query transform seems unnecessary. However, in order to avoid a certain problematic behavior of our model we will introduce something quite similar to a query transform; the details will be provided later. As before, we will take string length to be uniformly distributed on $1 \ldots M-1$; the parameter space Θ will be the word simplex $I\!P_{\mathcal{V}}$.

5.7.3 Link detection algorithm

Our approach to Link Detection will be as follows. Given the partial news stream $\mathcal{C}$ and the two stories $\mathbf{d}^1$ and $\mathbf{d}^2$ we will:

1. convert the stories $\mathbf{d}^1, \mathbf{d}^2$ into query-like representations $\mathbf{q}^1, \mathbf{q}^2$
2. use the news stream $\mathcal{C}$ to estimate posterior parameter vectors $E_{\mathbf{q}^1}u$ and $E_{\mathbf{q}^2}u$
3. compute a modified form of relative entropy between the resulting parameter vectors
4. if relative entropy falls below a certain threshold, conclude that $\mathbf{d}^1$ and $\mathbf{d}^2$ discuss the same topic; if entropy is above the threshold – stories talk about different events

The overall process is quite similar to the ad-hoc and relevance feedback scenarios. Nevertheless, the nature of TDT leads to a few interesting changes.

5.7.3.1 Transforming documents into queries

The first change involves transforming the original story $\mathbf{d}^1$ into a query-like representation $\mathbf{q}^1$; the same is done for $\mathbf{d}^2$. We are going to define a query generator that attempts to pick out the most salient key-words from $\mathbf{d}^1$. There exist a large number of IR heuristics that can help us achieve this goal. For example, from the standpoint of the 2-Poisson indexing model [52], we simply want to find *elite* words for $\mathbf{d}^1$. We are going to approach the problem in a slightly different fashion and use the hypergeometric distribution instead of the Poisson. To tease out the salient words we will ask the following question for every word v:

> *Suppose we randomly draw n words from the entire stream $\mathcal{C}$.*
> *What is the chance we will observe n_v instances of v?*

Here n is the length of $\mathbf{d}^1$, and n_v is the number of times the word v was observed in $\mathbf{d}^1$. We can answer the above question with a hypergeometric distributions, which assumes sampling from $\mathcal{C}$ without replacement.

[5] A possible exception to this claim is the tracking task with a large number of training stories. In that case it may be possible to improve performance in a way that cannot be traced back to improved link detection accuracy.

Let N_v denote the total number of times v occurs in the news stream, and take N to be the total number of words in the stream. The likelihood of getting n_v instances of v by pure chance is:

$$P_{chance}(n(v, \mathbf{d}^1)) = \left(\begin{array}{c} N_v \\ n_v \end{array} \right) \left(\begin{array}{c} N - N_v \\ n - n_v \end{array} \right) / \left(\begin{array}{c} N \\ n \end{array} \right) \tag{5.30}$$

In order to get the most salient words, we pick 10 words that have the *lowest* $P_{chance}(n_v)$, i.e. the query $\mathbf{q}^1$ will be composed of the words that are *least* likely to occur in $\mathbf{d}^1$ under the background statistics. The same procedure is applied to $\mathbf{d}^2$ to get the query $\mathbf{q}^2$.

There are two motivations for taking the admittedly heuristic step of collapsing a story to a handful of keywords. The first is computational complexity: reducing the number of words will lead to a significantly faster computation of parameter vectors $E_{\mathbf{q}^1}u$ and $E_{\mathbf{q}^2}u$. The number of query words forms a bottleneck in the algorithmic optimizations we use to compute equation (5.6): estimating a posterior for a 1000-word document takes 100 times longer than a similar posterior for a 10-word query. Computational complexity is all the more important because in link detection we have to repeat the estimation tens of thousands of times – one for each story in each testing pair. Contrast that with previous scenarios, which involved at most 100-200 queries each.

The second motivation comes from the observations we made in the relevance feedback scenario (section 5.2). Recall that long observations lead to a particularly simple expression for the expected parameter vector $E_{\mathbf{q}}u$ – the form given by equation (5.14). In our case, multiplying together thousands of probabilities (one for every word in $\mathbf{d}^1$) would make the posterior density $p(du|\mathbf{d}^1)$ spiked over a single point $u_{\mathbf{d}^1}$. As a result, the parameter vector $E_{\mathbf{d}^1}u$ will be identical to the empirical distribution of words in $\mathbf{d}^1$. Informally, this means that our estimation efforts have no effect – we end up with the original document $\mathbf{d}^1$, with none of the *synonymy* or *query expansion* effects that gave us a boost in ad-hoc retrieval performance. This is clearly undesirable. Reducing the story $\mathbf{d}^1$ to a small set of keywords $\mathbf{q}^1$ allows us to steer around the problem. The posterior density $p(du|\mathbf{q}^1)$ will be spread out over more points in $I\!P_{\mathcal{V}}$, and the resulting estimate will be a mixture of many related stories. The story $\mathbf{d}^1$ itself will be highly weighted element of this mixture, the important point is that it will not be the *only* element.

5.7.3.2 Modified relative entropy

After reducing each story to its keyword representation we can compute the expected parameter vectors $E_{\mathbf{q}^1}u$ and $E_{\mathbf{q}^2}u$ the same way as in the adhoc scenario: using equation (5.6). The next step is to measure the dissimilarity of the two vectors. Relative entropy is a natural measure of dissimilarity for distributions, and it served us well in the previous scenarios. However, link detection is different from earlier tasks in one very important respect: it is a *symmetric* problem. In all previous cases we had a single query $\mathbf{q}$ and had to rank documents with respect to that query. In a way, the query served as a fixed point for the scoring procedure: one side in the relative entropy was always fixed. In link detection we have no query to act as a fixed point: both sides are allowed to vary. Our dissimilarity measure should be symmetric to reflect this fact. After much experimentation we settled on the following measure:

$$D_{lnk}(\mathbf{d}^1, \mathbf{d}^2) = \sum_{v \in \mathcal{V}} E_{\mathbf{q}^1}u(v) \log \frac{E_0 u(v)}{E_{\mathbf{q}^2}u(v)} + \sum_{v \in \mathcal{V}} E_{\mathbf{q}^2}u(v) \log \frac{E_0 u(v)}{E_{\mathbf{q}^1}u(v)} \tag{5.31}$$

An intuition for equation (5.31) is as follows. We start with the relative entropy $D(E_{\mathbf{q}^1}u||E_{\mathbf{q}^2}u)$, which gives the number of bits wasted in "encoding" story 1 using story 2. We normalize it by subtracting $D(E_{\mathbf{q}^1}u||E_0 u)$ – the number of bits wasted by "encoding" story 1 with the background distribution. The quantity $D(E_{\mathbf{q}^1}u||E_0 u)$ is meant to compensate for the overall difficulty of encoding story 1. In Information Retrieval the quantity is known as the *clarity* measure[37]; it is used to guess the degree of ambiguity inherent in the user's query. After subtracting the clarity we get $D(E_{\mathbf{q}^1}u||E_{\mathbf{q}^2}u) - D(E_{\mathbf{q}^1}u||E_0 u)$, which is the number of bits wasted by encoding story 1 with story 2 as opposed to the background distribution. Now we can repeat the argument in the opposite direction, encoding story 2 with story 1, and comparing to the background. The final step is to add the relative number of bits wasted in encoding story 1 and then story 2 – the left and right summations of equation (5.31).

The last part of the link detection process is to compare the dissimilarity measure $D_{lnk}(\mathbf{d}^1, \mathbf{d}^2)$ to a threshold value. If dissimilarity exceeds the threshold we conclude that the stories discuss different events. If dissimilarity is below the threshold, we decide that $\mathbf{d}^1$ and $\mathbf{d}^2$ talk about the same event.

5.7.4 Experiments

In this section we evaluate performance of relevance models, as described above, on the Link Detection task of TDT. First, we describe the experimental setup and the evaluation methodology. Then we provide empirical support for using a modified form of relative entropy. Finally we show that relevance models significantly outperform simple language models, as well as other heuristic techniques. The experiments discussed in this section were previously reported in [2, 72].

5.7.4.1 Datasets and Topics

The experiments in this section were performed on three different datasets: TDT2, TDT3 and TDT4 [24]. Most of the training was done on a 4-month subset of the TDT2 dataset. The corpus contains 40,000 news stories totaling around 10 million words. The news stories were collected from six different sources: two newswire sources (Associated Press and New York Times), two radio sources (Voice of America and Public Radio International), and two television sources (CNN and ABC). The stories cover January through April of 1998. Radio and television sources were manually transcribed at closed-caption quality. Testing was carried out on the TDT3 and TDT4 datasets, containing 67,111 and 98,245 news stories respectively. The total number of words was around 21 million for TDT3 and 39 million for TDT4. Both datasets contain English, Arabic and Chinese stories as newswire, radio and television reports. In a pre-processing stage, all English stories were stemmed using a dictionary-based stemmer, and 400 stop-words from the InQuery [3] stop-list were removed.

TDT is concerned with detecting and organizing the topics in news stories. Human annotators identified a total of 96 topics in the TDT2 dataset, ranging from 1998 Asian financial crisis, to the Monica Lewinsky scandal, to an execution of Karla Faye Tucker. Each topic is centered around a specific event, which occurs in a specific place and time, with specific people involved. 56 out of these 96 topics are sufficiently represented in the first four months of 1998 and will form the basis for our evaluation. Similar annotations were done on the TDT3 and TDT4 datasets.

5.7.4.2 Evaluation Paradigm

The system is evaluated in terms of its ability to detect the pairs of stories that discuss the same topic. A total of 6363 story pairs were drawn from the dataset (according to the official TDT2 sampling). 1469 of these were manually [24] judged to be on-target (discussing the same topic), and 4894 were judged off-target (discussing different topics). During evaluation the Link Detection System emits a YES or NO decision for each story pair. If our system emits a YES for an off-target pair, we get a False Alarm error; if the system emits a NO for on-target pair, we get a Miss error. Otherwise the system is correct. Link Detection is evaluated in terms of the decision cost [45], which is a weighted sum of probabilities of getting a Miss and False Alarm:

$$Cost = P(Miss) \cdot C_{Miss} + P(FA) \cdot C_{FA}$$

In current evaluations of Link Detection, C_{Miss} is typically set to 10×0.02, and $C_{FA} = 1 \times 0.98$. Note that by always answering YES a system would have no misses and therefore a cost of 0.2 (similarly, always answering NO guarantees a cost of 0.98). To penalize systems for doing no better than a simple strategy like that, the cost is normalized by dividing by the minimum of those two values (here, 0.2). A normalized cost value near or above 1.0 reflects of a system of marginal value. An operational Link Detection System requires a threshold selection strategy for making YES / NO decisions. However, in a research setting it has been a common practice to ignore on-line threshold selection and perform evaluations at the threshold that gives the best possible cost. All of our experiments report the minimum normalized detection cost: $Cost_{min}$.

> *Before we proceed we would like to repeat that TDT is evaluated using a* cost *metric, not an accuracy metric. In all TDT experiments* **lower means better**.

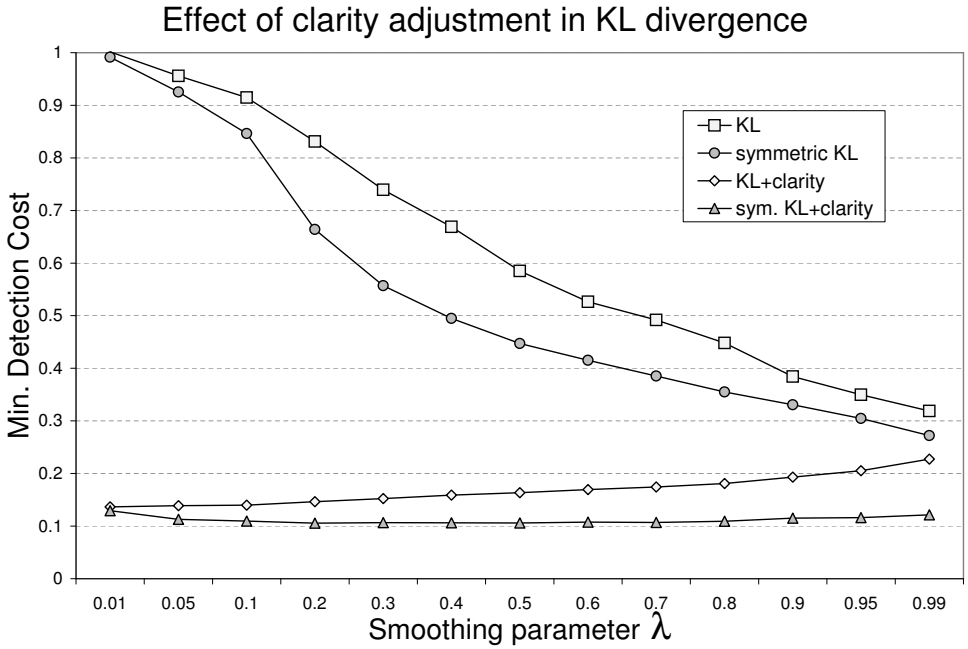


Figure 5.13. Link Detection performance on the TDT2 dataset. Normalization leads to significantly lower error rates. Symmetric versions of KL perform better than asymmetric versions. Symmetric KL with normalization is best and most stable with respect to the smoothing parameter λ.

5.7.4.3 Evaluation of various entropy formulations

In Section 5.7.3.2 we described a modified relative entropy measure, and provided an argument for why we believe this measure may perform better than straight relative entropy. To evaluate the value of our modification we perform a simple experiment without constructing relevance models. Given a pair of stories A and B we construct empirical language models of each story, smooth them with the background model U, and measure divergence. We consider four different divergence measures:

- relative entropy (KL divergence): $KL_1(A,B) = D(A||B)$
- symmetric version of divergence: $KL_2(A,B) = D(A||B) + D(B||A)$
- normalized relative entropy: $KL_{c,1}(A,B) = D(A||B) - D(A||U)$
- symmetric normalized entropy: $KL_{c,2}(A,B) = KL_{c,1}(A,B) + KL_{c,1}(B,A)$

Figure 5.13 shows the minimum detection cost ($Cost_{min}$) of the four measures as a function of the smoothing parameter λ from equation (4.28). We observe that clarity-adjusted KL leads to significantly lower errors for all values of λ. Clarity normalization also leads to smaller dependency on λ, which makes tuning easier. We also note that for both simple and normalized KL, we get significantly better performance by using symmetric divergence. The best performance $Cost_{min} = 0.1057$ is achieved by using the symmetric version of normalized KL when $\lambda = 0.2$. This performance will be used as a baseline in later comparisons with relevance models. The baseline is competitive with the state-of-the-art results reported in the official TDT evaluations [99].

5.7.4.4 Performance on the TDT2 / TDT3 datasets

Experiments discussed in this section were performed for the 2001 TDT evaluation and were originally reported in [72]. That evaluation used a slightly different technique for converting stories into queries. Instead of using the hypergeometric distribution to pick 10 *most unusual* words, in [72] we simply picked 30 *most frequent* words from each document. Overall detection cost was similar, but the procedure was approximately

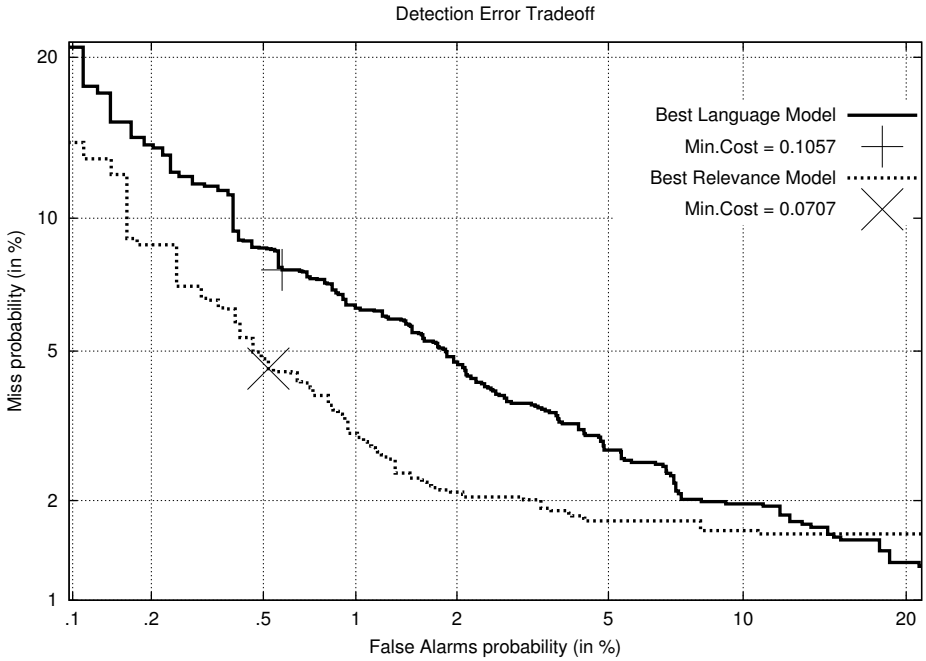


Figure 5.14. Relevance model noticeably outperforms the baseline for all threshold settings in the region of interest. Minimum Detection Cost is reduced by 33%

five times more expensive computationally. We compare a simple language modeling baseline to the performance we can achieve with relevance models. Given a pair of stories A and B, we construct a relevance model from each story as described in section 5.7.3.1. We use symmetric normalized KL as a measure of divergence between the two resulting relevance models. The smoothing parameter λ is set to 0.999.

Figure 5.14 shows the *Detection Error Tradeoff* (DET) [83] curve for the performance of relevance models, compared to the best language modeling baseline. A DET curve is a plot of Miss and False Alarm probabilities as a function of a sliding threshold. The point on each curve marks the optimal threshold, and the corresponding minimum cost $Cost_{min}$. Note that the values are plotted on a Gaussian scale and that the axes only go up to 20% Miss and False Alarm; the full-range DET curves are presented in Figure 5.15. We observe that a relevance modeling system noticeably outperforms the baseline for almost all threshold settings. The improvements are particularly dramatic around the optimal threshold. The minimum cost is reduced by 33%. Outside of the displayed region, on the high-precision end ($FalseAlarm < 0.01\%$), the relevance modeling system noticeably outperforms the baseline. On the very high-recall end ($Miss < 1.5\%$), the baseline performs somewhat better.

The results in Figure 5.14 were achieved by careful tuning of parameters on the 4-month subset of the TDT-2 corpus and do not represent a blind evaluation. However, the same parameters were used in the official TDT 2001 blind evaluation on the 3-month TDT-3 corpus. The results in that case were comparable to those described above (see Figure 5.16 for details). The system based on Relevance Models significantly outperformed a state-of-the-art vector-space system (cosine with Okapi *tf.idf* weighting). The normalized minimum cost was reduced from 0.27 to 0.24. This suggests that our parameter settings generalized reasonably well to the new dataset and the new set of topics.

5.7.4.5 Performance on the TDT3 / TDT4 datasets

We now turn our attention to performance of our model in the official TDT-2003 evaluation. Figure 5.17 shows the performance of the relevance model compared to the performance of our baseline system: vector-space, using cosine similarity with tf·idf weighting. Experiments were done on the 2003 training corpus (TDT3). Each pair of documents was compared in its own language when possible, all cross-language pairs were compared in machine-translated English. We observe that relevance model noticeably outperforms the

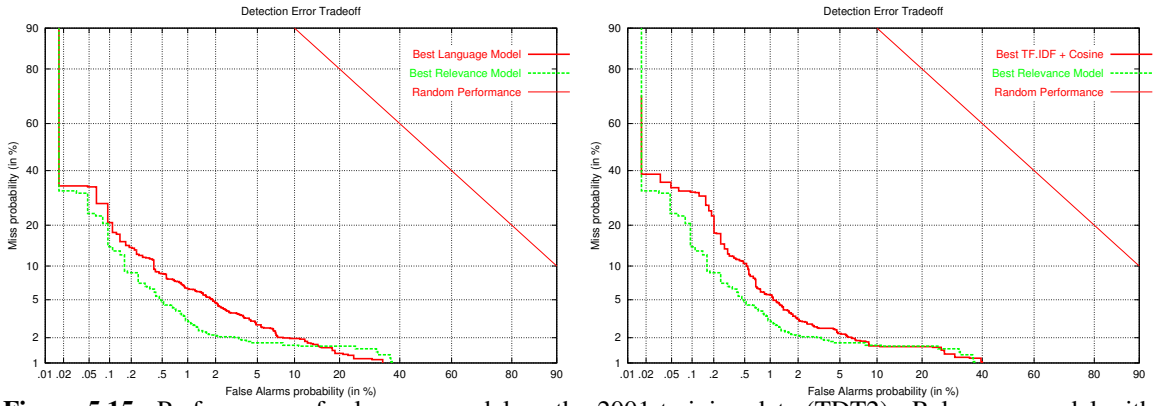


Figure 5.15. Performance of relevance model on the 2001 training data (TDT2). Relevance model with optimal parameters outperforms both the optimal Language Modeling system (left), and the optimal vector-space system using Cosine with Okapi term weighting (right). Minimum Detection Cost was reduced by 33% and 25% respectively (not shown).

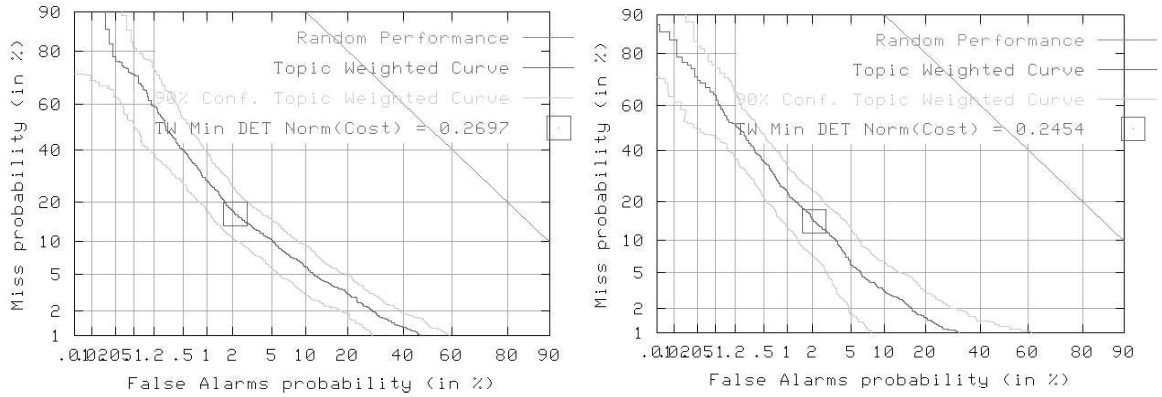


Figure 5.16. Performance of relevance models in the official TDT 2001 evaluation (TDT3). All the parameters were tuned on the training dataset, and no part of the evaluation dataset was used prior to evaluation. Relevance model (right) consistently outperforms the vector-space model (left). Minimum Detection Cost is reduced by 10%.

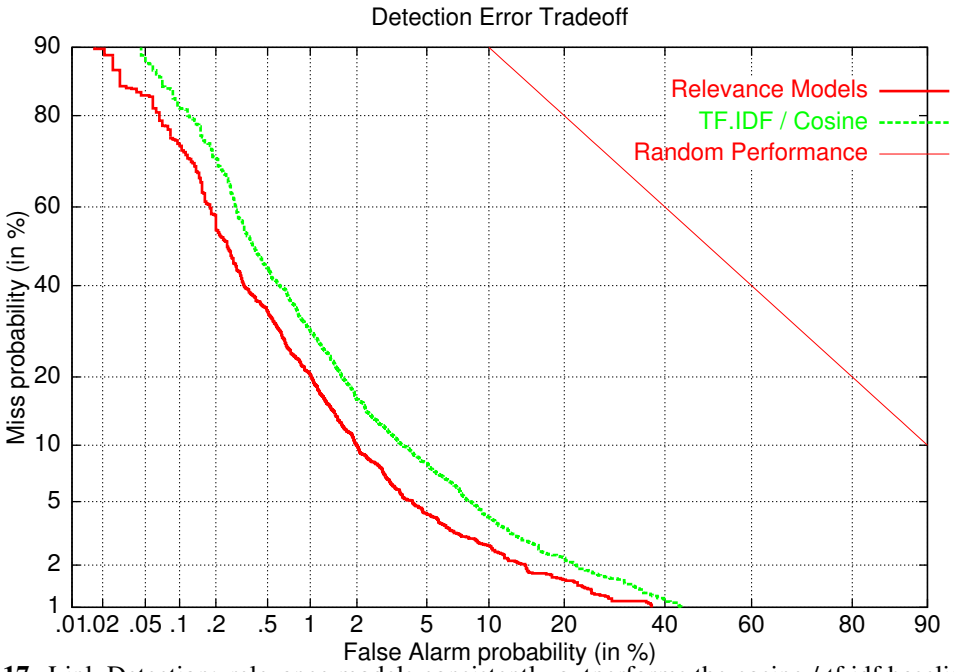


Figure 5.17. Link Detection: relevance models consistently outperforms the cosine / tf·idf baseline on the 2003 training data (TDT3).

vector-space baseline at all Miss levels. Table 5.14 provides a more detailed comparison of performance on both TDT-3 and TDT-4 datasets (pre-adjudicated results). We show results of using both a 1 database (where every pair is compared using machine-translated English), and 4 databases (where mono-lingual pairs are compared in their own language). We report the minimum cost for TDT-3 and both minimum and actual cost for TDT-4 (where we had to guess the optimal threshold).

	Minimum Cost		Cost
Detection Algorithm	TDT-3	TDT-4	TDT-4
Cosine / tf·idf (1db)	0.3536	0.2472	0.2523
relevance model (1db)	0.2774	—	—
Cosine / tf·idf (4db)	0.2551	0.1983	0.2000
relevance model (4db)	0.1938	0.1881	0.1892
rel.model + ROI (4db)	**0.1862**	**0.1863**	**0.1866**

Table 5.14. Performance of different algorithms on the Link Detection task. Comparing in native language (4db) is a definite win. Relevance models substantially outperforms the tf·idf baseline. Rule of Interpretation (ROI[2]) has a marginal effect. Boldface marks minimum cost in each column.

We can draw several conclusions from the results in Table 5.14. First, we observe that performing comparisons in the native language (4db) is always significantly better than comparing everything in machine-translated English (1db). This holds both for the vector-space baseline and for the relevance modeling approach. Second, the relevance model consistently outperforms the baseline both on TDT-3 and TDT-4. The relative improvement in minimum cost is a very substantial 27% on TDT-3 data, but only 7% on TDT-4, suggesting that smoothing parameters of the model were over-tuned to the training data. Our official submission to the TDT 2003 evaluation is represented by the last line in the table. *ROI* [2] refers to an attempt to further improve performance by adjusting story similarities depending on which *rule of interpretation* [24] they fall into. The adjustment has a very small but positive effect on performance. We observe that our threshold selection strategy for TDT-4 was quite successful: actual cost is quite close to the minimum cost. Bold figure

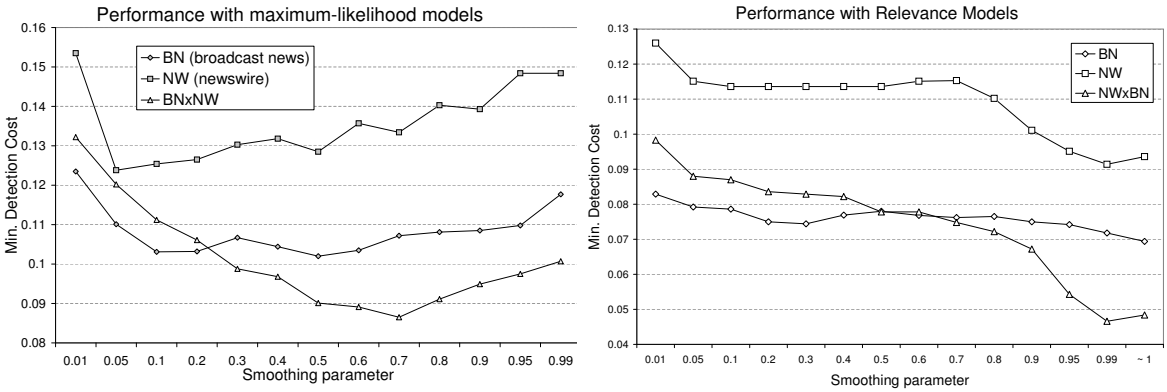


Figure 5.18. Performance on different source conditions. Left: baseline, optimal smoothing value is different for every condition. Right: relevance models, all conditions are optimized as λ approaches 1.

mark the best run in each column. It is worth noting that in addition to improved performance, the current formulation of the relevance model for Link Detection is almost 80% faster than the formulation described in [72]. The computational speedup is mainly due to using a small subset of words as a "query".

5.7.4.6 Cross-modal evaluation

An essential part of TDT is being able to deal with multiple sources of news. The TDT2 corpus that was used in our experiments includes news from six different sources. Two of these (Associated Press and New York Times) are printed sources, the other represent broadcast news, which are transcribed from audio signal. Spoken text has very different properties compared to written sources, and part of the TDT challenge is development of the algorithms that can cope with source-related differences in reporting. To determine how well our algorithms perform on different source conditions, we partitioned the set of 6363 pairs into three subsets:

1. 2417 pairs where both stories come from a broadcast source; this set will be labeled "BN" (broadcast news)
2. 1027 pairs where both stories come from a printed source; this set will be labeled "NW" (newswire)
3. 2919 pairs where one story is from a broadcast source and the other from the printed source; this set will be labeled "NWxBN"

Figure 5.18 shows performance of the baseline and the relevance modeling systems on the three subsets we described. Performance is shown as a function of the smoothing parameter λ. First we observe that performance varies very significantly from one subset to another. Interestingly, both systems perform best on the "NWxBN" condition, even though it intuitively appears to be more challenging as we are dealing with two different language styles. Another very interesting issue is the value of λ that gives the best performance. Note that for the baseline system the optimal λ value is different for every condition: "BN" is optimized near $\lambda = 0.5$, "NW" – near $\lambda = 0.05$, while "NWxBN" is optimal near $\lambda = 0.7$. This means that for the baseline system we cannot select a single value of λ which will work well for all sources. In contrast to that, for the relevance modeling system all conditions are optimized if we set λ to 0.99, or any value close to 1. This is a very encouraging result, as it shows that relevance models are not very sensitive to source conditions.

CHAPTER 6

CONCLUSION

Our sincere hope is that a reader will walk away from this thesis with a new way of thinking about relevance. We started this thesis by asserting that relevance serves as the cornerstone of Information Retrieval; we should perhaps have also said that relevance is the greatest stumbling block for the field. The fundamental task of information retrieval is to *retrieve relevant items in response to the query*. The main challenge lies in the fact that relevance is never observed. We know the query, we have the information items, but in most retrieval scenarios we will never be able to directly observe relevance, at least not until *after* we have already retrieved the items. In some sense this makes the fundamental retrieval task ill-defined. After all, how can we ever learn the concept of relevance if we have no examples to learn from? The answer is: we cannot, not unless we make some assumptions. These assumptions typically fall into two categories. In the first category we follow the classical probabilistic model [113] and come up with heuristics to approximate relevance. In the second category, e.g. language modeling [102], we get rid of relevance and develop a formal model of retrieval based on some hypothesized process that generates queries from documents, or vice versa. The goal of this thesis was to provide a third alternative.

We introduced a model of retrieval that treats relevance as a generative process underlying *both* documents and queries. The centerpiece of our model is the generative relevance hypothesis, which states that for a given information need, relevant documents and relevant queries can be viewed as random samples from the same underlying model. To our knowledge, none of the existing retrieval models are based on a similar assumption. The generative hypothesis allows us to formally and naturally tie together relevance, documents and user queries. By treating both documents and queries as observable samples from the unknown relevance model, we are able to estimate all relevant probabilities without resorting to heuristics. In addition, assuming a common generative model allowed us to provide effective solutions for a large number of diverse retrieval problems. The generative approach allows us to associate words with pictures, video frames, and other words in different languages.

If theoretical arguments are not convincing, we would like to point out that the proposed model is very effective from an empirical standpoint. We have demonstrated that our generative model meets or exceeds state-of-the-art performance in the following retrieval scenarios:

1. **Ad-hoc retrieval.** Our model provides a formal approach to ad-hoc retrieval. We compared our generative model to four strong baselines: InQuery tf·idf [3], tf·idf with LCA [147], language models [102] and language models with query expansion [101]. Comparisons were carried out on five different datasets: WSJ, AP, FT, LA, TDT2. The datasets contain a total of over 800,000 documents and 450 queries. Our model outperforms the standard tf·idf baseline by 15-25% on all datasets. Most of the improvements are statistically significant. The model exceeds performance of the other three baselines by 5-10%.

2. **Relevance feedback.** The process of relevance feedback is very natural in our model: relevant examples are treated as samples from the underlying relevance model – in the same way as the query. We compared performance of our generative model to language models with relevance feedback [101]. Experiments were repeated on three different datasets (AP,FT,LA) and involved a total of 250 queries. The generative model performed slightly better than the baseline.

3. **Cross-language retrieval.** The proposed generative model can easily learn to associate words in different languages. We described an effective way to retrieve Chinese documents in response to English queries. We tested our model on the TREC-9 dataset with 25 queries. We were able to outperform the state-of-the-art translation model [150] by 10% on long and short queries. Our model achieved 96-98% of the strong mono-lingual performance.

4. **Handwriting retrieval.** We demonstrated that our model can associate English words with manuscripts that are not susceptible to OCR. Our model created the first operational system for searching manuscript collections with text queries. Our model achieves over 80% precision at rank 1 on a collection of George Washington's handwritings. Mean average precision is 54%, 63%, 78% and 89% for 1-, 2-, 3- and 4-word queries respectively. There are no baselines we could compare to.

5. **Image retrieval.** A continuous-space extension of our model represents the current best-performing system for automatic annotation of images with keywords. We tested the model on the Corel benchmark dataset [41] and found it to outperform the best published results by up to 60%. The model provides a very effective means for retrieving unlabeled images in response to multi-word textual queries. Mean average precision is 23%, 25%, 31% and 44% for 1-, 2-, 3- and 4-word queries respectively.

6. **Video retrieval.** The same continuous-space model facilitates text-based searching through larger collections of video footage. In our experiments the model achieves mean average precision of 30%, 26%, and 32% for 1-, 2-, and 3-word queries respectively.

7. **Topic Detection and Tracking.** Last, but not least, the generative model results in a very significant improvement in the accuracy of link detection – the core technology behind TDT. Under in-house evaluations, our model yields a very substantial 25-30% improvement over the best baseline systems. More importantly, our model outperforms the same baselines by 5-7% in completely blind annual evaluations carried out by NIST. Our model was the best performing link detection system in the 2003 TDT evaluation.

The second contribution of this thesis is a new and effective generative process for modeling discrete exchangeable sequences (bags of features). In our opinion, this process may be of interested to a much wider audience, beyond the broad field of Information Retrieval. Nothing in the model is specific to language, to documents and queries – it can be applied whenever one is trying to learn something about unordered discrete observations, e.g. feature sets. Our formulation was motivated in no small part by the work of Blei, Ng and Jordan [12]. We view their work as a very important development in discrete generative models, but we strongly disagree with the structural assumptions made in their LDA model, the same structural assumptions that previously led Hoffman to develop the PLSI model.

Specifically, we believe it is a bad idea to represent text collections in terms of a small number of factors. Natural language is an incredibly complex phenomenon, and in our experience it is best to think of it as a myriad of outliers, rather than a handful of clusters and some Gaussian noise. Any type of dimensionality reduction, be it clustering, information bottleneck methods, principal component analysis, mixture models, PLSI or LDA will invariably destroy rare events, which are so important for users.

As we mentioned before, we do not argue with the notion that the *dimensionality* of text is lower than the number of words in the vocabulary. We believe it is quite plausible that documents in a given collection really do form a low-dimensional sub-space. However, we strongly disagree with the conception that this sub-space may be linear, or even convex. It is our personal view that collections of documents are stringy, fibrous, contagious in nature: an author writes a story, then someone picks it up and writes a response, the author revises the story, then someone discovers new facts relevant to it. The original story morphs and evolves in a step-by-step fashion, each step being taken not too far the predecessor, but after a number of steps we may end up quite far from where we started. We believe it highly unlikely that the majority of these paths will revolve around a certain point in space – the point which we as humans would like to call the *topic* of discussion.

We wanted to create a model that would be able to capture these paths in their original form, rather than cramming them into a low-dimensional polytope defined by cluster centroids, principal components or PLSI aspects. To that end we proposed a kernel-based density allocation model, which can be thought of as a series of spheres, placed along the paths along which the stories develop. In our opinion, the model is simple and effective. It is very easy to train, and easily allows for discriminative training when the goal is to achieve a good performance on a certain specific task. The most important quality of the model is that it makes no structural assumptions about the data. Every training example is preserved and has a say when we predict the likelihood of new events. We have devoted chapter 5 to showing that this kernel-based approach is very effective in seven very different retrieval scenarios.

Finally, in the course of writing this dissertation we uncovered a number of very curious new facts about existing models. These facts are:

- contrary to popular belief, the classical probabilistic model is not based on the assumption of word independence. Instead, it requires a much more plausible assumption of *proportional interdependence* (section 2.3.2.5).
- there is a good reason why explicit models of word dependence have never resulted in consistent improvements in retrieval performance, either in the classical or the language-modeling framework (sections 2.3.2.5 and 2.3.3.4).
- documents and queries can be viewed as samples from the same underlying distribution, even though they may look very different (sections 3.2.2.2 and 3.2.2.3).
- document likelihood, or probability ratio, is not the best criterion for ranking documents; query likelihood is a better alternative (section 3.2.3.5).
- a ganerative probabilistic LSI model with k aspects is equivalent to a regular, document-level mixture model with $N >> k$ components; the original PLSI is equivalent to an "oracle" unigram model (section 4.3.4).
- Latent Dirichlet Allocation is not a word-level mixture, it is a regular document-level mixture. One can think of it as a simple Dirichlet model restricted to the k-topic sub-simplex (section 4.3.5).
- one should be very cautious when analyzing graphical models, as they can be quite misleading (section 4.3.6).

6.1 Limitations of our Work

We can gain further understanding of our model by looking at the cases that cannot be handled with our methods. We will highlight three limitations that we consider to be the most important: **(i)** the closed-world nature of our retrieval model, **(ii)** the assumption of exchangeability, and **(iii)** the computational cost of our model.

6.1.1 Closed-universe approach

Our model of Information Retrieval represents a closed-universe approach. We assume that there exists a single information need $\mathcal{R}$, along with an associated relevance model $P_{\mathcal{R}}(\cdot)$. According to the generative hypothesis, each relevant document is a sample from $P_{\mathcal{R}}(\cdot)$, while each non-relevant document is a sample from some other probability distribution. This works well as long as we have only one information need. But suppose we are dealing with two distinct information needs $\mathcal{R}_1$ and $\mathcal{R}_2$, and some document d happens to be relevant to both of them. In this case we would run into a philosophical inconsistency: according to the GRH, d must have been drawn from $P_{\mathcal{R}_1}(\cdot)$, since it is relevant to $\mathcal{R}_1$. But d is also relevant to $\mathcal{R}_2$, so it must be an observation from $P_{\mathcal{R}_2}(\cdot)$, which implies it cannot be a sample from $P_{\mathcal{R}_1}(\cdot)$, unless the two relevance models are identical. We would like to point out that the same kind of inconsistency is present in the classical probabilistic model [113]. Furthermore, Robertson and Hiemstra [107] argue that an even greater inconsistency lies at the foundation of the language-modeling framework: assuming that the query q is a sample drawn from a relevant document implies that there may be only one relevant document in the entire collection, namely the document d^* that was the source of q.

The important thing to realize that the inconsistencies described above are purely philosophical. They have absolutely no practical impact since we are dealing with *probabilities* that a given sample (d or q) would be drawn from a given language model, and not with hard yes-or-no decisions about the origin of the sample. If we insist on a philosophically consistent model, the solution is to define a new universe (event space) for every new information need $\mathcal{R}$. The same closed-universe approach is characteristic of all existing probabilistic retrieval models, with a possible exception of the *unified* model [14, 105].

6.1.2 Exchangeable data

Our generative model is based on the assumption that representation components (words) $X_1 \ldots X_n$ are exchangeable. As we discussed in section 3.2.2, the assumption of exchangeability is much weaker than the popular assumption of word independence. However, it is an assumption about the data, and thus constitutes a limitation of our model. Our model can only be applied in the cases where we can reasonably assume that the order of features (words) is irrelevant. In chapter 5 we demonstrated that the assuming exchangeability works for a surprisingly wide array of possible tasks, but we can certainly imagine scenarios where the model would not perform well.

A good example of a domain where our model might fail is *music retrieval*. Suppose we have a collection of musical pieces, where each piece is expressed as a sequence of notes. The semantics of each musical piece is embodied not in the individual notes, but rather in their *sequence*. Assuming exchangeability of notes in a piece would prove disastrous: individual notes are completely meaningless when taken out of context. It would be equivalent to assuming that individual *characters* in text are exchangeable.

Another domain where our model may not fare well is hierarchical data. Suppose we want to perform retrieval tasks on a *treebank* corpus, where individual documents represent parse trees. Leaves of a given parse tree are not as sequence-dependent as notes in a musical pieces: in many grammars we can re-order the sub-trees and still retain the meaning. However, we would lose a lot of information if we assumed the leaves to be completely exchangeable.

Fortunately, in many practical scenarios we can construct an exchangeable representation that can approximate order-dependent data to some degree. In music retrieval we may consider using n-gram sequences rather than individual notes; exchangeability of these n-grams is a more realistic assumption. Similarly, in the case of hierarchical data, we would consider the entire path from a root to the leaf as a single "word", and then assume that these paths are exchangeable.

6.1.3 Computational complexity

A serious limitation of our model is the relatively high computational expense associated with kernel-based estimates. As we discussed in section 4.4.3, the number of parameters in our model grows linearly with the size of the training set. These are not free parameters, so overfitting is not a grave concern, but the sheer number of them makes the model inefficient relative to the baselines. For certain scenarios the added computational expense is not justified by a relatively small improvement in performance. For example, we report 20-25% improvement in average precision for the ad-hoc retrieval scenario. While these improvements are substantial and statistically significant, they do not represent a groundbreaking leap in retrieval accuracy: we may gain an extra relevant document in the top 10 results. At the same time, the computational expense of our generative model is an order of magnitude higher than the expense of a simple word-matching baseline. In this case a 25% improvement in precision is probably not worth the added cost.

In other retrieval cases, the situation is quite different. For example, in the image retrieval scenario there are no simple and effective baselines. Any operational system has to be able to associate bitmap images with textual captions, which is not a trivial process. State-of-the-art baselines are based on statistical translation and maximum-entropy models. The combined (training + testing) complexity of these models is comparable to the overall expense of using our generative model. And even if the computational cost was higher, the 60% improvement in accuracy would certainly justify the added expense.

6.2 Directions for Future Research

Our work is far from complete. The ideas presented in this thesis can be extended in a number of important ways, and applied in many scenarios beyond the ones we described. In the remainder of this chapter we will briefly outline the directions that we consider most promising, the directions that we hope to explore in our future work.

6.2.1 Semi-structured and relational data

All of the applications presented in chapter 5 involved retrieval of un-structured entities. We believe that our approach has natural extensions to semi-structured and relational environments.

Consider a very simple case of relational structure: *hyperlinks*. Every document d in our collection may have outgoing links to some set of documents $d' \in \eta_d$. The set η_d may be called the set of *neighbors* of d. Is there a way we can model this sort of structure within our generative model? We believe that we can, and the extension is surprisingly straightforward. Let $L_1 \ldots L_k$ denote the k links going out of a document. Each L_i is a multinomial random variable, its possible values are the identifiers of documents in our collection. For example, the event $L_1 = 23$ would mean that the first out-going link points to the twenty-third document in the collection. The set of all possible identifiers forms a finite vocabulary $\mathcal{L}$. At a conceptual level, this vocabulary is no different from a vocabulary of a real language. We can pretend that $L_1 \ldots L_k$ represent words in some foreign language, perhaps Lithuanian, and use the same approach that was described in section 5.3. Given a set of hyper-linked documents we can learn the associations between the English words and the outgoing links. Then, given a new un-linked document, we can attempt to predict which existing documents it might point to. Note that the same procedure can be used to learn associations between the content of a document and links pointing *into* it. In other words, we should be able to predict which documents are likely to point into the new item. Equivalently, given a set of incoming links, we will be able to predict a set of English words that are likely to occur in the page.

6.2.1.1 Relational structures

Once we are able to model the simple link structure, the extension to relational data is very natural. The distinguishing feature of relational databases is the fact that each link has a *type* associated with it. For example, consider a university personnel database. Each entity represents a person, and these entities can be linked in a number of different ways. We may have *course_instructor* links going from a professor to a large group of students, as well as *academic_advisor* links going to a much smaller group. We can also have *shared_office* links between the students and *shared_grant* links between the professors. We can model relational links associated with each person by a set of random variables $L_1 \ldots L_k$. Each variable L_i takes values in the finite space $\mathcal{L} \times \mathcal{T}$, where $\mathcal{L}$ is the set of entity identifiers and $\mathcal{T}$ is the set of all possible link types. Conceptually, $\mathcal{L} \times \mathcal{T}$ is just another vocabulary, and we can directly apply the cross-language techniques from section 5.3 to learn associations between the attributes of persons and their inter-relationships.

6.2.1.2 Semi-structured data

The idea of learning associations between entity attributes and the relational structure is certainly not novel. Researchers in the data mining community have proposed dozens of very effective and efficient algorithms for relational learning. We have no a-priori reason to believe that our model will outperform any of the existing solutions. However, our model does have one very important advantage over the classical mining algorithms. Most of these algorithms rely on *fully-structured* data representations. In order to learn that a certain person A is likely to have a *shared_office* link with person B we need to have access to entity attributes, such as age, academic position, research interests, etc. Classical mining algorithms will not work if instead of these attributes we provide them with the personal webpages of A and B, which are likely to be *unstructured* narratives. Our generative model has an advantage because it was specifically designed for learning associations from unstructured data. In fact, as far as our model is concerned, every aspect of the data is unstructured, some aspects just happen to have a very constrained vocabulary. Learning the presence of a *shared_office* link is no different than learning the presence of some Chinese word: both can be conditioned on a free-form English narrative. In a way, our model presents an interesting alternative to the popular mining approach: instead of *information extraction* followed by *data mining* we would directly learn associations over a mix of structured and unstructured components.

6.2.2 Order-dependent data

We believe it is possible to extend the generative model described in chapter 4 to order-dependent data. Consider a simple case of trying to learn a first-order dependence model, similar to a bi-gram. We have a

training collection of observed bigrams $\mathcal{C}_{train} = \{(a_i, b_i) : i{=}1 \ldots N\}$. The goal is to estimate the conditional probability distribution $P(a|b)$, representing the likelihood of observing a given that the previous word was b. A popular approach to estimating this distribution involves taking the maximum-likelihood estimate $\frac{\#(a,b)}{\#(b)}$, backing off to the unigram model and further smoothing the unigram counts. We suspect that a more powerful estimate may be constructed by setting $P(a|b) = \frac{P(a,b)}{\sum_a P(a,b)}$, where the distribution $P(a, b)$ is interpreted as a joint probability of seeing an Arabic word a together with a Bulgarian word b. The joint probability would be estimated as described in section 5.3 with $\mathcal{C}_{train}$ playing the role of a parallel corpus. The main advantage of the new estimate would be its ability to handle possible heterogeneity of word sequences.

6.2.3 Dirichlet kernels

In section 4.4 we described two types of kernel for our allocation model: the Dirac delta kernel and the Dirichlet kernel. Considerations of computational efficiency forced us to do most of our experiments with the simpler delta kernels. However, results reported in section 4.5 show that Dirichlet kernels represent a much more accurate instrument for modeling textual data. We have a strong interest in exploring the potential benefits of Dirichlet kernels in real retrieval scenarios. Explorations of this kind will require us to come up with a new set of algorithmic optimizations to bring the efficiency of our Dirichlet implementation a bit closer to the efficiency of a delta-based implementation.

BIBLIOGRAPHY

[1] Allan, J. *Introduction to Topic Detection and Tracking*. Kluwer Academic Publishers, Massachusetts, 2002, pp. 1–16.

[2] Allan, J., Bolivar, A., Connell, M., Cronen-Townsend, S., Feng, A., Feng, F., Kumaran, G., Larkey, L., Lavrenko, V., and Raghavan, H. UMass TDT 2003 research summary. In *NIST Topic Detection and Tracking Workshop* (Gaithersburg, MD, November 2003).

[3] Allan, J., Connell, M., Croft, W. B., Feng, F. F., Fisher, D., and Li, X. INQUERY and TREC-9. In *Proceedings of the Ninth Text REtrieval Conference (TREC-9)* (2000), pp. 551–562.

[4] Allan, J., Gupta, R., and Khandelval, V. Temporal summaries of news topics. In *Proceedings of the 24th International ACM SIGIR Conference on Research and Development in Information Retrieval* (New Orleans, LA, September 2001), pp. 10–18.

[5] Allan, J., J.Carbonell, G.Doddington, J.Yamron, and Y.Yang. Topic detection and tracking pilot study: Final report. In *Proceedings of DARPA Broadcast News Transcription and Understanding Workshop* (1998), pp. 194–218.

[6] Allan, J., Jin, H., Rajman, M., Wayne, C., Gildea, D., Lavrenko, V., Hoberman, R., and Caputo, D. Topic-based novelty detection. Clsp workshop final report, Johns Hopkins University, 1999.

[7] Allan, J., Lavrenko, V., and Jin, H. First story detection in TDT is hard. In *Proceedings of the Ninth International Conference on Information and Knowledge Management CIKM* (Wasnihgton, DC, 2000), pp. 374–381.

[8] Barnard, K., Duygulu, P., Freitas, N., Forsyth, D., Blei, D., and Jordan, M. Matching words and pictures. *Journal of Machine Learning Research 3* (2003), 1107–1135.

[9] Belkin, N. J., Oddy, R. N., and Brooks, H. M. ASK for information retrieval: Part i. background and theory. *Journal of Documentation 38*, 2 (1982), 61–71.

[10] Belkin, N. J., Oddy, R. N., and Brooks, H. M. ASK for information retrieval: Part ii. results of a design study. *Journal of Documentation 38*, 3 (1982), 145–164.

[11] Berger, A., and Lafferty, J. Information retrieval as statistical translation. In Hearst et al. [54], pp. 222–229.

[12] Blei, D. M., Ng, A. Y., and Jordan, M. I. Latent dirichlet allocation. In *Technical Report UCB//CSD-02-1194* (August 2002).

[13] Blei, D. M., Ng, A. Y., and Jordan, M. I. Latent dirichlet allocation. *Journal of Machine Learning Research 3* (2003), 993–1022.

[14] Bodoff, D., and Robertson, S. E. A new unified probabilistic model. *Journal of the American Society for Information Science 55*, 6 (2003), 471–487.

[15] Bookstein, A., and Swanson, D. Probabilistic models for automatic indexing. *Journal of the American Society for Information Science 25*, 5 (1974), 312–319.

[16] Bookstein, A., and Swanson, D. A decision theoretic foundation for indexing. *Journal of the American Society for Information Science 26* (1975), 45–50.

[17] Brown, P. F., Cocke, J., Della Pietra, S. A., Della Pietra, V. J., Jelinek, F., Lafferty, J. D., Mercer, R. L., and Roossin, P. S. A statistical approach to machine translation. *Computational Linguistics 16*, 2 (1990), 79–85.

[18] Bruza, P. D. *Stratified information disclosure: A synthesis between hypermedia and information retrieval*. PhD dissertation, University of Nijmegen, Nijmegen, The Netherlands, 1993.

[19] Bruza, P. D., and van der Weide, T. P. The modeling and retrieval of documents using index compression. *SIGIR Forum 25*, 2 (1991), 91–103.

[20] Bruza, P. D., and van der Weide, T. P. Stratified hypermedia structures for information disclosure. *The Computer Journal 35*, 3 (1992), 208–220.

[21] Carbonell, J., and Goldstein, J. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval* (1998).

[22] Carson, C., Thomas, M., Belongie, S., Hellerstein, J.M., and Malik, J. Blobworld: A system for region-based image indexing and retrieval. In *Proceedings of the Third International Conference on Visual Information Systems* (1999), pp. 509–516.

[23] Chen, Stanley F., and Goodman, Joshua T. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th Annual Meeting of the ACL* (1996).

[24] Cieri, C., Strassel, S., Graff, D., Martey, N., Rennert, K., and Liberman, M. *Corpora for Topic Detection and Tracking*. Kluwer Academic Publishers, Massachusetts, 2002, pp. 33–66.

[25] Cooper, W. A definition of relevance for information retrieval. *Information Storage and Retrieval 7*, 1 (1971), 19–37.

[26] Cooper, W. Exploiting the maximum entropy principle to increase retrieval effectiveness. *Journal of the American Society for Information Science 34*, 2 (1983), 31–39.

[27] Cooper, W., and Huizinga, P. The maximum entropy principle and its application to the design of probabilistic retrieval systems. *Information Technology Research and Development 1*, 2 (1982), 99–112.

[28] Cooper, W., and Maron, M. Foundation of probabilistic and utility-theoretic indexing. *Journal of the Association for Computing Machinery 25*, 1 (1978), 67–80.

[29] Cooper, W. S. Some inconsistencies and misidentified modeling assumptions in probabilistic information retrieval. *ACM Transactions on Information Systems 13*, 1 (January 1995), 100–111.

[30] Cosijn, E., and Ingwersen, P. Dimensions of relevance. *Information Processing and Management 36* (2000), 533–550.

[31] Crestani, F., and Van Rijsbergen, C. J. Information retrieval by logical imaging. *Journal of Documentation 51*, 1 (1995), 3–17.

[32] Crestani, F., and Van Rijsbergen, C. J. Probability kinematics in information retrieval. In *Proceedings of the 18th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Seattle, WA, 1995), pp. 291–299.

[33] Croft, W. B., Ed. *Advances in Information Retrieval — Recent Research from the Center for Intelligent Information Retrieval*, vol. 7 of *The Kluwer international series on information retrieval*. Kluwer Academic Publishers, Boston, MA, April 2000.

[34] Croft, W. B., and Harper, D. Using probabilistic models of document retrieval without relevance information. *Journal of Documentation 35* (1979), 285–295.

[35] Croft, W. B., Harper, D. J., Kraft, D. H., and Zobel, J., Eds. *Proceedings of the Twenty-Fourth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval* (New Orleans, LA, September 2001), ACM Press.

[36] Croft, W. B., and Van Rijsbergen, C. J. An evaluation of Goffman's indirect retrieval method. *Information Processing ad Management 12*, 5 (1976), 327–331.

[37] Cronen-Townsend, S., Zhou, Y., and Croft, W. B. Predicting query performance. In *In the proceedings of the 25th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Tampere, Finland, August 2002), pp. 299–306.

[38] Cuadra, C., and Katter, R. Opening the black box of "relevance". *Journal of Documentation 23*, 4 (1967), 291–303.

[39] Cuadra, C., and Katter, R. The relevance of relevance assessment. *Proceedings of the American Documentation Institute 4* (1967), 95–99.

[40] Dumais, S. Latent semantic indexing (lsi): Trec-3 report. In *TREC-3 Proceedings* (Gaithersburg, Maryland, November 1994), pp. 219–230.

[41] Duygulu, P., Barnard, K., Freitas, N., and Forsyth, D. Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. In *Proceedings of the Seventh European Conference on Computer Vision* (2002), pp. 97–112.

[42] Eisenberg, M. *Magnitude estimation and the measurement of relevance*. PhD dissertation, Syracuse University, Syracuse, NY, 1986.

[43] Fairthorne, R. A. *Implications of test procedures*. Case Western Reserve University Press, Cleveland, 1963, pp. 109–113.

[44] Feng, S., Manmatha, R., and Lavrenko, V. Multiple bernoulli relevance models for image and video annotation. In *Proceedings of the International Conference on Pattern Recognition (CVPR)* (Washington, DC, July 2004).

[45] Fiscus, Jonathan G., and Doddington, George R. *Topic Detection and Tracking Evaluation Overview*. Kluwer Academic Publishers, Massachusetts, 2002, pp. 17–31.

[46] Foskett, D. J. Classification and indexing in the social sciences. *Proceedings of ASLIB 22* (1970), 90–100.

[47] Foskett, D. J. A note on the concept of relevance. *Information Storage and Retrieval 8*, 2 (1972), 77–78.

[48] Goffman, W. An indirect method of information retrieval. *Information Storage and Retrieval 4* (1969), 361–373.

[49] Harman, D. Overview of the TREC 2002 novelty track. In *Proceedings of TREC 2002 (notebook)* (2002), pp. 17–28.

[50] Harper, D. J. *Relevance feedback in document retrieval systems*. PhD thesis, University of Cambridge, UK, February 1980.

[51] Harper, D. J., and van Rijsbergen, C. J. An evaluation of feedback in document retrieval using co-occurrence data. *Journal of Documentation 34* (1978), 189–216.

[52] Harter, S. P. A probabilistic approach to automatic keyword indexing. *Journal of the American Society for Information Science 26*, 4 and 5 (1975), Part I: 197–206; Part II: 280–289.

[53] Harter, S. P. Variations in relevance assessments and the measurement of retrieval effectiveness. *Journal of the American Society for Information Science 47* (1996), 37–49.

[54] Hearst, M., Gey, F., and Tong, R., Eds. *Proceedings of the Twenty-Second Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval* (Berkeley, CA, August 1999), ACM Press.

[55] Hearst, M., and Pedersen, J. Re-examining the cluster hypothesis: scatter/gather on retrieval results. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval* (1996), pp. 76–84.

[56] Hiemstra, D. *Using Language Models for Information Retrieval*. PhD dissertation, University of Twente, Enschede, The Netherlands, January 2001.

[57] Hiemstra, D., and de Jong, F. Disambiguation strategies for cross-language information retrieval. In *Proceedings of the Third European Conference on Research and Advanced Technology for Digital Libaries, ECDL'99*, S. Abiteboul and A.-M. Vercoustre, Eds., vol. 1696 of *Lecture Notes in Computer Science*. Springer-Verlag, Paris, September 1999, pp. 274–293.

[58] Hoffmann, T. Probabilistic latent semantic indexing. In *Proceedings on the 22nd annual international ACM SIGIR conference* (1999), pp. 50–57.

[59] Ingwersen, P. *Information Retrieval Interaction*. Taylor Graham, London, 1992.

[60] Janes, J. The binary nature of continuous relevance judgments: A case study of users' perceptions. *Journal of the American Society for Information Science 42*, 10 (1991), 754–756.

[61] Janes, J. On the distribution of relevance judgments. In *Proceedings of the American Society for Information Science* (Medford, NJ, 1993), pp. 104–114.

[62] Jardine, N., and Van Rijsbergen, C. J. The use of hierarchical clustering in information retrieval. *Information Storage and Retrieval 7* (1971), 217–240.

[63] Jeon, J., Lavrenko, V., and Manmatha, R. Automatic image annotation and retrieval using cross-media relevance models. In *In the proceedings of the 26th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Toronto, Canada, August 2003), pp. 119–126.

[64] Kantor, P. Maximum entropy and the optimal design of automated information retrieval systems. *Information Technology Research and Development 3*, 2 (1984), 88–94.

[65] Katter, R. The influence of scale form on relevance judgment. *Information Storage and Retrieval 4*, 1 (1968), 1–11.

[66] Lafferty, J., and Zhai, C. Document language models, query models, and risk minimization for information retrieval. In Croft et al. [35], pp. 111–119.

[67] Lafferty, J., and Zhai, ChengXiang. *Probabilistic relevance models based on document and query generation*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 2003, pp. 1–10.

[68] Lalmas, M., and Van Rijsbergen, C. J. A logical model of information retrieval based on situation theory. In *Proceedings of the BCS 14th Information Retrieval Colloquium* (Lancaster, UK, 1992), pp. 1–13.

[69] Lalmas, M., and Van Rijsbergen, C. J. Situation theory and Dempster-Shafer's theory of evidence for information retrieval. In *Proceedings of Workshop on Incompleteness and Uncertainty in Information Systems* (Concordia University, Montreal, Canada, 1993), pp. 62–67.

[70] Lalmas, M., and Van Rijsbergen, C. J. Information calculus for information retrieval. *Journal of the American Society for Information Science 47* (1996), 385–398.

[71] Lancaster, F. W. *Information retrieval systems: Characteristics, testing and evaluation*. John Wiley and Sons, New York, 1979.

[72] Lavrenko, V., Allan, J., DeGuzman, E., LaFlamme, D., Pollard, V., and Thomas, S. Relevance models for topic detection and tracking. In *Proceedings of Human Language Technologies Conference, HLT 2002* (2002), pp. 104–110.

[73] Lavrenko, V., and Croft, W. B. *Relevance Models in Information Retrieval*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 2003, pp. 11–56.

[74] Lavrenko, V., Feng, S., and Manmatha, R. Statistical models for automatic video annotation and retrieval. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (Montreal, Canada, May 2004).

[75] Lavrenko, V., Manmatha, R., and Jeon, J. A model for learning the semantics of pictures. In *Proceedings of the Seventeenth Annual Conference on Neural Information Processing Systems (NIPS)* (Vancouver, Canada, December 2003).

[76] Lavrenko, V., Rath, T., and Manmatha, R. Holistic word recognition for handwritten historical documents. In *Preceedings of the International Workshop on Document Image Analysis for Libraries (DIAL)* (Palo Alto, CA, January 2004).

[77] Lee, J., and Kantor, P. A study of probabilistic information retrieval systems in the case of inconsistent expert judgment. *Journal of the American Society for Information Science 42*, 3 (1991), 166–172.

[78] Lenat, D. CYC: A large scale investment in knowledge infrastructure. *Communications of the ACM 38*, 11 (1995), 33–38.

[79] Leuski, A. Evaluating document clustering for interactive information retrieval. In *Proceedings of CIKM 2001 conference* (2001), pp. 33–40.

[80] Tipster Volume 1. CDROM available from Linguistic Data Consortium, University of Pennsylvania, 1993, Revised March 1994. http://morph.ldc.upenn.edu/Catalog/LDC93T3B.html.

[81] Manmatha, R., and Croft, W. B. *Word spotting: Indexing handwritten manuscripts*. AAAI/MIT Press, Dordrecht, The Netherlands, 1997, pp. 43–64.

[82] Maron, M. E., and Kuhns, J. L. On relevance, probabilistic indexing and information retrieval. *Journal of the Association for Computing Machinery 7*, 3 (1960), 216–244.

[83] Martin, A., Doddington, G., Kamm, T., and Ordowski, M. The DET curve in assessment of detection task performance. In *EuroSpeech* (1997), pp. 1895–1898.

[84] Meghini, C., Sebastiani, F., Straccia, U., and Thanos, C. A model of information retrieval based on terminological logic. In *Proceedings of the 16th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Pittsburgh, PA, 1993), pp. 298–307.

[85] Metzler, D., Lavrenko, V., and Croft, W. B. Formal multiple-Bernoulli models for language modeling. In *Proceedings of the 27th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Sheffield, UK, July 2004).

[86] Miller, D., Leek, T., and Schwartz, R. BBN at TREC7: Using hidden markov models for information retrieval. In Voorhees and Harman [142], pp. 133–142.

[87] Miller, D. R. H., Leek, T., and Schwartz, R. M. A hidden markov model information retrieval system. In Hearst et al. [54], pp. 214–221.

[88] Mizarro, S. Relevance: The whole history. *Journal of the American Society for Information Science 48*, 9 (1997), 810–832.

[89] Mizarro, S. How many relevances in information retrieval? *Interacting with Computers 10*, 3 (1998), 305–322.

[90] Mori, Y., Takahashi, H., and Oka, R. Image-to-word transformation based on dividing and vector quantizing images with words. In *Proceedings of the First International Workshop on Multimedia Intelligent Storage and Retrieval Management MISRM'99* (1999).

[91] Nallapati, R., and Allan, J. Capturing term dependencies using a sentence tree based language model. In *Proceedings of CIKM 2002 conference* (2002), pp. 383–390.

[92] Nallapati, R., and Allan, J. An adaptive local dependency language model: Relaxing the naive Bayes assumption. In *SIGIR'03 Workshop on Mathematical Methods in Information Retrieval* (Toronto, Canada, 2003).

[93] Nallapati, R., Croft, W. B., and Allan, J. Relevant query feedback in statistical language modeling. In *Proceedings of CIKM 2003 conference* (2003), pp. 560–563.

[94] Nie, J. Y. An outline of a general model for information retrieval. In *Proceedings of the 11th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Grenoble, France, 1988), pp. 495–506.

[95] Nie, J. Y. An information retrieval model based on modal logic. *Information Processing and Management 25*, 5 (1989), 477–491.

[96] Nie, J. Y. Towards a probabilistic modal logic for semantic-based information retrieval. In *Proceedings of the 15th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Copenhagen, Denmark, 1992), pp. 140–151.

[97] Nie, J. Y., Brisebois, M., and Lepage, F. Information retrieval is counterfactual. *The Computer Journal 38*, 8 (1995), 643–657.

[98] Nigam, K., McCallum, A., Thrun, S., and Mitchell, T. Text classification from labeled and unlabeled documents using em. *Machine Learning Journal 39*, 2/3 (2000), 103–134.

[99] NIST. Proceedings of the tdt 2001 workshop. Notebook publication for participants only, Nov. 2001.

[100] Papka, R. *On-line New Event Detection, Clustering, and Tracking*. PhD dissertation, TR99-45, University of Massachusetts, Amherst, MA, 1999.

[101] Ponte, J. M. *A language modeling approach to information retrieval*. Phd dissertation, University of Massachusets, Amherst, MA, September 1998.

[102] Ponte, J. M., and Croft, W. B. A language modeling approach to information retrieval. In *Proceedings of the Twenty-First Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval* (Melbourne, Australia, August 1998), W. B. Croft, A. Moffat, C. J. van Rijsbergen, R. Wilkinson, and J. Zobel, Eds., ACM Press, pp. 275–281.

[103] Rath, T., Lavrenko, V., and Manmatha, R. A statistical approach to retrieving historical manuscript images. In *CIIR Technical Report MM-42* (2003).

[104] Robertson, S. On Bayesian models and event spaces in information retrieval. Presented at SIGIR 2002 Workshop on Mathematical/Formal Models in Information Retrieval, 2002.

[105] Robertson, S. The unified model revisited. Presented at SIGIR 2003 Workshop on Mathematical/Formal Models in Information Retrieval, 2003.

[106] Robertson, S., and Bovey, J. Statistical problems in the application of probabilistic models to information retrieval. Technical Report 5739, British Library Research and Development Department, 1991.

[107] Robertson, S., and Hiemstra, D. Language models and probability of relevance. In *Proceedings of the Workshop on Language Modeling and Information Retrieval* (Pittsburgh, PA, May 2001), pp. 21–25.

[108] Robertson, S., and Walker, S. On relevance weights with little relevance information. In *Proceedings of the 20th International ACM SIGIR Conference on Research and Development in Information Retrieval* (1997), pp. 16–24.

[109] Robertson, S. E. The probabilistic character of relevance. *Information Processing and Management 13* (1977), 247–251.

[110] Robertson, S. E. The probability ranking principle in IR. *Journal of Documentation 33* (1977), 294–304. Reprinted in [129].

[111] Robertson, S. E., Maron, M. E., and Cooper, W. S. Probability of relevance: a unification of two competing models for document retrieval. *Information Technology: Research and Development 1* (1982), 1–21.

[112] Robertson, S. E., Maron, M. E., and Cooper, W. S. *The unified probabilistic model for IR*. Springer-Verlag, Berlin, Germany, 1983, pp. 108–117.

[113] Robertson, S. E., and Sparck Jones, K. Relevance weighting of search terms. *Journal of the American Society for Information Science 27* (1976), 129–146. Reprinted in [145].

[114] Robertson, S. E., and Walker, S. Some simple effective approximations to the 2–poisson model for probabilistic weighted retrieval. In *Proceedings of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval* (Dublin, Ireland, July 1994), W. B. Croft and C. J. van Rijsbergen, Eds., Springer-Verlag, pp. 232–241.

[115] Robertson, S. E., Walker, S., and Beaulieu, M. M. Okapi at TREC-7: automatic ad hoc, filtering, VLC and interactive track. In Voorhees and Harman [142], pp. 253–264.

[116] Saracevic, T. Relevance: A review of and a framework for the thinking on the notion in information science. *Journal of the American Society for Information Science 26*, 6 (1975), 321–343.

[117] Saracevic, T. Relevance reconsidered'96. In *Information Science: Integration in Perspective* (Royal School of Library and Information Science, Copenhagen, Denmark, 1996), pp. 201–218.

[118] Sebastiani, F. A probabilistic terminological logic for information retrieval. In *Proceedings of the 17th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Dublin, Ireland, 1994), pp. 122–130.

[119] Sebastiani, F. On the role of logic in information retrieval. *Information Processing and Management 38*, 1 (1998), 1–18.

[120] Shi, J., and Malik, J. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 22*, 8 (2000), 888–905.

[121] Silverman, B. *Density Estimation for Statistics and Data Analysis*. CRC Press, 1986, pp. 75–94.

[122] Song, F., and Croft, W. B. A general language model for information retrieval. In *Proceedings of Eighth International Conference on Information and Knowledge Management, CIKM'99* (1999).

[123] Song, F., and Croft, W. B. A general language model for information retrieval. In *Proceedings of the Twenty-Second Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval, SIGIR'99* (1999), pp. 279–280.

[124] Sparck Jones, K. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation 28* (1972), 11–21. Reprinted in [145].

[125] Sparck Jones, K. LM vs. PM: Where's the relevance? In *Proceedings of the Workshop on Language Modeling and Information Retrieval* (Pittsburgh, PA, May 2001), pp. 12–15.

[126] Sparck Jones, K., Robertson, S., Hiemstra, D., and Zaragoza, H. *Language Modeling and Relevance*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 2003, pp. 57–72.

[127] Sparck Jones, K., Walker, S., and Robertson, S. A probabilistic model of information retrieval: development and comparative experiments. part 1. *Information Processing and Management 36* (2000), 779–808.

[128] Sparck Jones, K., Walker, S., and Robertson, S. A probabilistic model of information retrieval: development and comparative experiments. part 2. *Information Processing and Management 36* (2000), 809–840.

[129] Sparck Jones, K., and Willett, P., Eds. *Readings in information retrieval*. Multimedia Information and Systems. Morgan Kaufmann, San Francisco, CA, 1997.

[130] Turtle, H. R. *Inference Network for Document Retrieval*. PhD dissertation, University of Massachusets, Amherst, MA, 1990.

[131] Turtle, H. R., and Croft, W. B. Evaluation of an inference network-based retrieval model. *ACM Transactions on Information Systems 9*, 3 (July 1991), 187–222.

[132] van Rijsbergen, C. J. A theoretical basis for the use of co-occurrence data in information retrieval. *Journal of Documentation 33* (1977), 106–119.

[133] van Rijsbergen, C. J. *Information retrieval*, second ed. Butterworth & Co (Publishers) Ltd, London, UK, 1979.

[134] Van Rijsbergen, C. J. A new theoretical framework for information retrieval. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval* (Pisa, Italy, 1986), p. 200.

[135] Van Rijsbergen, C. J. A non-classical logic for information retrieval. *The Computer Journal 29* (1986), 481–485.

[136] van Rijsbergen, C. J. A non-classical logic for information retrieval. *The Computer Journal 29* (1986), 481–485.

[137] Van Rijsbergen, C. J. Towards an information logic. In *Proceedings of the 12th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Cambridge, MA, 1989), pp. 77–86.

[138] Van Rijsbergen, C. J., and Croft, W. B. Document clustering: An evaluation of some experiments with the Cranfield 1400 collection. *Information Processing and Management 11* (1975), 171–182.

[139] Vickery, B. C. The structure of information retrieval systems. *Proceedings of the International Conference on Scientific Information 2* (1959), 1275–1290.

[140] Vickery, B. C. Subject analysis for information retrieval. *Proceedings of the International Conference on Scientific Information 2* (1959), 855–865.

[141] Voorhees, E. *The effectiveness and efficiency of agglomerative hierarchic clustering in document retrieval*. PhD dissertation, TR85-705, Cornell University, NY, 1986.

[142] Voorhees, E. M., and Harman, D. K., Eds. *Proceedings of the Seventh Text REtrieval Conference (TREC-7)* (Gaithersburg, MD, November 1998), National Institute of Standards and Technology (NIST) and Defense Advanced Research Projects Agency (DARPA), Department of Commerce, National Institute of Standards and Technology.

[143] Voorhees, E. M., and Harman, D. K., Eds. *Proceedings of the Ninth Text REtrieval Conference (TREC-9)* (Gaithersburg, MD, November 2000), Department of Commerce, National Institute of Standards and Technology.

[144] Walpole, Ronald E., and Myers, Raymond H. *Probability and Statistics for Engineers and Scientists*. MacMillan Publishing Company, New York, 1989, pp. 108–109.

[145] Willett, P., Ed. *Document Retrieval Systems*, vol. 3 of *Foundations of Information Science*. Taylor Graham, London, UK, 1988.

[146] Wilson, P. Situational relevance. *Information Storage and Retrieval 9*, 8 (1973), 457–471.

[147] Xu, J., and Croft, W. B. Query expansion using local and global document analysis. In *Proceedings of the Nineteenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval* (Zurich, Switzerland, August 1996), H.-P. Frei, D. Harman, P. Schäuble, and R. Wilkinson, Eds., ACM Press, pp. 4–11.

[148] Xu, J., and Croft, W. B. Cluster-based language models for distributed retrieval. In *Proceedings of the 22nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 99)* (Berkeley, CA, August 1999), pp. 15–19.

[149] Xu, J., and Weischedel, R. TREC-9 cross-lingual retrieval at BBN. In Voorhees and Harman [143], pp. 106–116.

[150] Xu, J., Weischedel, R., and Nguyen, C. Evaluating a probabilistic model for cross-lingual information retrieval. In Croft et al. [35], pp. 105–110.

[151] Yu, C., Buckley, C., and Salton, G. A generalized term dependence model in information retrieval. *Information Technology Research and Development 2* (1983), 129–154.

[152] Zaragoza, H., Hiemstra, D., and Tipping, M. Bayesian extension to the language model for ad-hoc information retrieval. In *In the proceedings of the 26th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Toronto, Canada, July 2003), pp. 4–9.

[153] Zhai, C. *Risk Minimization and Language Modeling in Text Retrieval*. PhD dissertation, Carnegie Mellon University, Pittsburgh, PA, July 2002.

[154] Zhai, C., and Lafferty, J. A study of smoothing methods for language models applied to ad hoc information retrieval. In Croft et al. [35], pp. 334–342.

[155] Zhai, C., and Lafferty, J. Two-stage language models for information retrieval. In *Proceedings of the Twenty-Fifth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval* (Tampere, FInland, 2002), pp. 49–56.